REDUCTION ALGORITHMS FOR THE CRYPTANALYSIS OF LATTICE BASED ASYMMETRICAL CRYPTOSYSTEMS

A Thesis Submitted to
the Graduate School of Engineering and Sciences of
İzmir Institute of Technology
in Partial Fulfillment of the Requirements for the Degree of

MASTER OF SCIENCE

in Computer Software

by
Mutlu BEYAZIT

August 2008
İZMİR

We approve the thesis of **Mutlu BEYAZIT**

Assoc. Prof. Dr. Ahmet KOLTUKSUZ
Supervisor

Asst. Prof. Dr. Gökhan DALKILIÇ
Committee Member

Dr. Serap ATAY
Committee Member

13 August 2008

Prof. Dr. Sıtkı AYTAÇ
Head of the Computer Engineering Department

Prof. Dr. Hasan BÖKE
Dean of the Graduate School of Engineering and Sciences

ACKNOWLEDGEMENTS

To begin with, I would like to thank Dr. Ahmet Koltuksuz for providing me with the opportunity to perform such a study.

Furthermore, I would like to thank my colleague Selma Tekir for pushing me forward to complete my thesis and her endurance.

I would also like to thank my former roommate Ali Mersin. If he were here, this thesis would not be finished (:D).

The (other) IS3 Lab. members, Burcu Kulahçıoğlu, Murat Özkan, Sevgi Uslu, Evren Akalp, Hüseyin Hışıl and Dr. Serap Atay, deserves my sincere thanks, too. It is a pleasure to work with them.

I would also like to thank my fellow research assistants, Deniz Çokuslu, Fatih Tekbacak and Orhan Dağdeviren for providing me with the latex templates.

Finally, I would like to thank my parents, Zahide and Yıldırım Beyazıt, my brother, Kutlu Beyazıt, and my girlfriend, Esra Aycan, for their consideration and understanding.

ABSTRACT

REDUCTION ALGORITHMS FOR THE CRYPTANALYSIS OF LATTICE BASED ASYMMETRICAL CRYPTOSYSTEMS

The theory of lattices has attracted a great deal of attention in cryptology in recent years. Several cryptosystems are constructed based on the hardness of the lattice problems such as the shortest vector problem and the closest vector problem. The aim of this thesis is to study the most commonly used lattice basis reduction algorithms, namely Lenstra Lenstra Lovasz (LLL) and Block Kolmogorov Zolotarev (BKZ) algorithms, which are utilized to approximately solve the mentioned lattice based problems. Furthermore, the most popular variants of these algorithms in practice are evaluated experimentally by varying the common reduction parameter *delta* in order to propose some practical assessments about the effect of this parameter on the process of basis reduction. These kind of practical assessments are believed to have non-negligible impact on the theory of lattice reduction, and so the cryptanalysis of lattice cryptosystems, due to the fact that the contemporary nature of the reduction process is mainly controlled by the heuristics.

ÖZET

KAFES TABANLI ASİMETRİK KRİPTOSİSTEMLERİN KRİPTANALİZİ İÇİN İNDİRGEME ALGORİTMALARI

Son yıllarda, kafes teorisi kriptolojide büyük ilgi görmektedir. En kısa vektör problemi ve en yakın vektör problemi gibi kafes tabanlı problemlerin zorluğuna dayanarak bir çok kriptosistem inşa edilmiştir. Bu tezin amacı bahsedilen kafes tabanlı problemleri yaklaşık olarak çözmek amacıyla en sık kullanılan kafes tabanı indirgeme algoritmalarından Lenstra Lenstra Lovasz (LLL) ve Block Kolmogorov Zolotarev (BKZ) algoritmalarının incelenmesidir. Bununla beraber, bu algoritmaların uygulamadaki en yaygın değişkelerinde ortak *delta* indirgeme parametresi değiştirilerek bu parametrenin indirgeme sürecindeki etkisi deneysel olarak incelenmekte ve taban indirgeme işlemini ilgilendiren uygulama temelli değerlendirmeler ortaya konmaktadır. Günümüz indirgeme işleminin doğasının ağırlıklı olarak buluşsal yöntemlerce kontrol edilmesi nedeniyle bu tip uygulama temelli değerlendirmelerin kafes indirgeme teorisinde, ve buna bağlı olarak kafes kriptosistemlerinin kriptanalizinde, ihmal edilemeyecek etkileri olduğuna inanılmaktadır.

TABLE OF CONTENTS

LIST OF FIGURES

Figure **Page**

CHAPTER 1

INTRODUCTION

Lattices are constructions of geometry which can be described as the set of intersection points of an infinite regular grid. Though the simplicity in their construction, lattices hide a rich complex structure which has attracted the interests of many mathematicians over the last two centuries. In the meantime, they have found plenty of applications in several branches of mathematics, especially in computer science. Their importance in cryptology is firstly noted with the knapsack based cryptosystems. Invention of the basis reduction algorithms and the breakthrough results of Ajtai on the hard problems over lattices triggered numerous researches on the theory of lattices, more specifically on the hard lattice problems and the lattice basis reduction algorithms. The aim of this study is to explore the particular lattice problems and their use in cryptology with a cryptanalytic perspective considering the practical algorithms to solve these problems.

The lattices are the source of many interesting computationally hard problems. Two such problems are the shortest vector and the closest vector problem. The shortest vector problem is by far the most popular lattice based problem and has many variations. This problem stems from the fact that every lattice has a non-zero shortest vector and the problem of finding such a vector is NP-hard. The impracticality of solving the shortest vector problem in its exact form leads to the emergence of approximate versions related to the problem. On the other hand, the closest vector problem is another important problem in the theory of lattices. It includes finding a lattice vector which is among the closest lattice vectors to an arbitrary target vector, or point, in the space. As in the case of the shortest vector problem, the closest vector problem is also NP-hard and has many variants.

The algorithms to solve the shortest vector problem can be classified as the exact algorithms and the approximate algorithms. The main exact algorithms are the HKZ (Hermite Korkine Zolotarev) reduction algorithms, (Kannan 1987b) and the (randomized) sieve algorithms, (Ajtai, et al. 2001). Whereas, the approximate algorithms are primarily the LLL (Lenstra Lenstra Lovasz), (Lenstra, et al. 1982), algorithms and the

BKZ (Block Korkine Zolotarev) algorithms, (Schnorr 1987). In practice, the exact algorithms are only practical in relatively low dimensions. Therefore, the approximation algorithms attract comparatively more interest. An interesting point on the approximation algorithms and on some of the exact algorithms is that, in general, they do more than just solving the approximate shortest vector problem, they produce a solution to the lattice basis reduction problem. Basically, they output a shorter and more orthogonal basis with particular properties, and this basis is deemed to be good in a practical point of view, because it can be used to (approximately) solve different lattice based problems. Another important point is that, all exact algorithms applies an approximation algorithm for preprocessing and the approximation algorithms usually make use of the exact algorithms by accessing them in low dimensions or relaxing their reduction conditions to achieve a better performance.

It is also possible to group the algorithms to solve the closest vector problem in two categories. The solution to the exact closest vector problem is mainly given by the algorithms presented in (Kannan 1987b, Blömer 2000, Klein 2000, Agrell, et al. 2002). The approximation to the closest vector problem is performed in a roundabout manner. The general approach is to use a reduction from the closest vector problem to an instance of the (approximate) shortest vector problem together with an algorithm to solve the reduced instance. This is a factor increasing the importance of the concept of the lattice basis reduction. To name some of them, the approximation algorithms can be found in (Babai 1986, Ajtai, et al. 2001, Blömer and Naewe 2007).

In this study, not all the algorithms to solve the referred lattice problems are mentioned but instead the basis reduction algorithms which carry relatively greater importance in the cryptanalysis of the lattice based public key cryptosystems are taken under the consideration. For this reason, the following algorithmics and related concepts are left out: the low dimensional reduction algorithms of Lagrange and Gauss , the algorithms of Hermite , and Korkine and Zolotarev , the HKZ reduction algorithms (Kannan 1983, 1987b, Helfrich 1985, Hanrot and Stehlé 2007, 2008), the sieve algorithms (Ajtai, et al. 2001, Schnorr 2001, Nguyen and Vidick 2008), the sampling algorithms (Ajtai, et al. 2002, Schnorr 2003, Ludwig 2005, Buchmann and Ludwig 2006, Blömer and Naewe 2007) and the closest vector algorithms (Kannan 1983, Babai 1986, Kannan 1987b, Schnorr 1996, Blömer 2000, Klein 2000, Ajtai, et al. 2001, Agrell, et al. 2002,

Blömer and Naewe 2007). Consequently, the LLL and the BKZ lattice basis reduction algorithms are focused upon.

First, mathematical background on the theory of lattices is introduced. The definition of lattices is given with the notion of lattice determinant, the Gram-Schmidt orthogonalization, the successive minima and the Hermite's constant. Furthermore, some important theorems, including Minkowski's theorems, which give initial bounds on the first successive minimum, are presented.

Second, the lattice problems of the interest (the shortest vector and the closest vector problems) are defined in their exact, decision, approximate and promise forms. Following this discussion, the complexity results on the shortest and the closest vector problems together with some important connections between them are mentioned in order to demonstrate the achieved theoretical hardness and the non-hardness bounds to solve or approximate these problems. In addition, the algorithmic results which are partially related to the practice are stated to exhibit the practical achievable bounds concerning the considered problems. The section is concluded after briefly introducing the lattice basis reduction problem.

Next, the LLL and the BKZ lattice basis reduction algorithms are outlined. In both cases, the theoretical infrastructure related to the algorithms is given by presenting the definition of reducedness with some extra notions, if required, and the worst-case achievable approximation factors are demonstrated. Furthermore, the algorithm is presented in its raw form in order to ease the theoretical exploration and the understanding, and a short analysis is performed on the running time and the correctness of the algorithm. In addition, the algorithm is rewritten in a contemporary manner, numerous variants are mentioned and some research is briefly pointed out. It is important to note that, the theory of the LLL algorithm is more simple when compared to that of the BKZ algorithms. Therefore, due to the technicality of the subject, no analysis on the running time or the correctness of the BKZ algorithms is performed for the sake of brevity.

Later, the importance of the basis reduction algorithms is discussed in a cryptanalytic perspective. The role of good basis in solving the shortest vector and the closest vector problems is demonstrated. In addition, a general cryptographic scheme emerging in the lattice based cryptosystems is outlined. Following this discussion, some heuristics which are useful in a cryptanalytic sense are presented and the section is concluded after

discussing the lattice based attacks on relatively more general lattice based cryptographic schemes.

Next, the most efficient variants of the LLL algorithm and the BKZ algorithm are considered for a practical assessment regarding a common runtime parameter belonging to the algorithms. It is important to note that, these variants are based on some certain heuristics and in general have no theoretical inspections. However, they are also the most preferred algorithms in practice and by the cryptanalysts due to their efficiency.

Finally, the study is concluded by presenting some remarks concerning this and possible future research.

CHAPTER 2

BACKGROUND

In this chapter, basic information on the lattices is conveyed. The fundamental definitions and important theorems related to the point lattices are presented keeping the discussion as simple and concise as possible.

2.1. Lattice and Basis

A lattice is the set of all integral combinations of n linearly independent vectors $\mathbf{b}_1, ..., \mathbf{b}_n \in \mathbb{R}^m$ denoted by

$$\Lambda = L(\mathbf{b}_1, \ldots, \mathbf{b}_n) = \left\{ \sum_{i=1}^{n} x_i \mathbf{b}_i : x_i \in \mathbb{Z}, \mathbf{b}_i \in \mathbb{R}^m \right\} \tag{2.1}$$

or

$$\Lambda = L(\mathrm{B}) = \left\{ \mathbf{B}\mathbf{x} : \mathbf{B} \in \mathbb{R}^{m \times n}, \mathbf{x} \in \mathbb{Z}^n \right\} \tag{2.2}$$

where $\mathrm{B} = \{\mathbf{b}_1, \ldots, \mathbf{b}_n\}$ is called the lattice basis and $\mathbf{B} = [\mathbf{b}_1 \ldots \mathbf{b}_n] \in \mathbb{R}^{m \times n}$ is the matrix representation of the basis with columns as the basis vectors. The integers n and m are called lattice dimension (or rank), $\dim(\Lambda)$, and lattice degree, $\deg(\Lambda)$, respectively, and when $n = m$ the lattice is called full dimensional. Without any references to a specific basis, lattice Λ can be defined to be a nonempty discrete subset of $\mathbb{R}^m$ which is closed under subtraction. Consequently, lattices are discrete additive subgroups of $\mathbb{R}^m$.

Linear span of the basis vectors is defined as

$$\mathrm{span}(\mathrm{B}) = \{\mathbf{B}\mathbf{x} : \mathbf{x} \in \mathbb{R}^n\}. \tag{2.3}$$

Since all the bases of the same lattice generate the same span, we can define the span of the lattice as $\mathrm{span}(\Lambda)$. It is obvious that

$$\dim(\Lambda) = \dim(\mathrm{span}(\Lambda)) \tag{2.4}$$

and $\mathrm{span}(\Lambda) = \mathbb{R}^m$ if and only if the lattice is full dimensional.

Half open parallelepiped associated with a lattice basis B is defined as

$$P(\mathrm{B}) = \left\{ \mathbf{Bx} : \mathbf{B} \in \mathbb{R}^{m \times n}, x_i \in [0,1) \right\} \tag{2.5}$$

It should be noted that if B' is another lattice basis then $P(\mathrm{B}')$ does not contain any lattice vectors other than zero.

Two lattice bases B and B' generate the same lattice if and only if there exist an integral matrix $\mathbf{U}$ with $|\det(\mathbf{U})| = 1$ such that $\mathbf{B}' = \mathbf{BU}$. $\mathbf{U}$ is also called unimodular matrix, and the bases B and B' are said to be equivalent. The statement can be proved using the half open parallelepipeds.

2.2. Determinant

For all $\mathbf{x}, \mathbf{y} \in \mathbb{R}^m$ and $a \in \mathbb{R}$, a norm is a function $\|.\| : \mathbb{R}^m \to \mathbb{R}$, which is generally used to measure length, such that

- It is positive definite, i.e. $\|\mathbf{x}\| > 0$ for $\mathbf{x} \neq \mathbf{0}$ and $\|\mathbf{0}\| = 0$.
- It is homogeneous, i.e. $\|a\mathbf{x}\| = |a| \cdot \|\mathbf{x}\|$.
- It satisfies the triangle inequality, i.e. $\|\mathbf{x}+\mathbf{y}\| \leqslant \|\mathbf{x}\| + \|\mathbf{y}\|$.

Furthermore, for any $p \geqslant 1$, the family of norm functions, l_p norms, are defined as

$$\|\mathbf{x}\|_p = \left(\sum_{i=1}^{m} |x_i|^p \right)^{1/p} \tag{2.6}$$

Note that l_2 norm is the Euclidean norm whereas l_∞ norm is called the maximum norm since

$$\|\mathbf{x}\|_\infty = \lim_{p \to \infty} \|\mathbf{x}\|_p = \max_{i=1}^{m} |x_i| . \tag{2.7}$$

Related to the norm distance between two vectors is

$$\mathrm{dist}(\mathbf{x}, \mathbf{y}) = \|\mathbf{x} - \mathbf{y}\| . \tag{2.8}$$

Gram-Schmidt orthogonalization process is almost always referred while studying the lattices. For any sequence of vectors $\mathbf{b}_1, ..., \mathbf{b}_n$, the Gram-Schmidt orthogonalized vectors are defined as

$$\mathbf{b}_i^* = \mathbf{b}_i - \sum_{j=1}^{i-1} \mu_{i,j} \mathbf{b}_j^* \tag{2.9}$$

where $\mu_{i,j} = \left\langle \mathbf{b}_i, \mathbf{b}_j^* \right\rangle / \left\langle \mathbf{b}_j^*, \mathbf{b}_j^* \right\rangle$ are called the Gram-Schmidt coefficients and $\langle ., . \rangle$ is the inner product. Note that $\mathbf{b}_i^*$ is the component of $\mathbf{b}_i$ orthogonal to span (B_{i-1}) where $\mathrm{B}_{i-1} = \{\mathbf{b}_1, \ldots, \mathbf{b}_{i-1}\}$ and span $(\mathrm{B}_i) =$ span (B_i^*), therefore the orthogonalized vectors depend on the order of the basis vectors. Since $\left\langle \mathbf{b}_i^*, \mathbf{b}_j^* \right\rangle = 0$ holds for all $i \neq j$, the set of vectors $\mathrm{B}^* = \{\mathbf{b}_1^*, ..., \mathbf{b}_n^*\}$ are pairwise orthogonal. Furthermore

$$\left\langle \mathbf{b}_i, \mathbf{b}_j^* \right\rangle = \begin{cases} 0 & i < j \\ \left\langle \mathbf{b}_j^*, \mathbf{b}_j^* \right\rangle & i = j \\ \mu_{i,j} \left\langle \mathbf{b}_j^*, \mathbf{b}_j^* \right\rangle & i > j \end{cases} \tag{2.10}$$

which consequently lead to the fact that $\mu_{i,j} = 0$ for $i < j$ and $\mu_{i,j} = 1$ for $i = j$. One should also note that, since not every lattice has a basis of mutually orthogonal vectors, although B^* is a basis for span (B), it is not necessarily a basis for the lattice $L(\mathrm{B})$.

Determinant of a lattice is defined as the volume of the fundamental parallelepiped

$$\det(\Lambda) = \mathrm{vol}(P(\mathrm{B})) \tag{2.11}$$

where $\Lambda = L(\mathrm{B})$. Equivalently we can define the determinant as

$$\det(L(\mathrm{B})) = \prod_{i=1}^{n} \|\mathbf{b}_i^*\| = \sqrt{\det(\mathbf{G})} \tag{2.12}$$

where $\mathbf{G} = \mathbf{B}^{\mathrm{T}}\mathbf{B}$ is the Gram matrix, i.e. $n \times n$ matrix whose (i, j)th entry equals $\mathbf{G}_{ij} = \left\langle \mathbf{b}_i, \mathbf{b}_j \right\rangle$. The determinant $\det(\Lambda)$ does not depend on any particular basis, i.e. for any two lattice bases B and B′, $\mathrm{vol}(P(\mathrm{B})) = \mathrm{vol}(P(\mathrm{B'}))$ is always true. The statement follows from the fact that lattice bases B and B′ are related by a unimodular matrix. Intuitively, as the volume of the fundamental parallelepiped $P(\mathrm{B})$ decreases, the density of the lattice points in span(B) increases.

2.3. Successive Minima and Hermite's Constant

Successive minima $\lambda = \lambda_1, \ldots, \lambda_n$ are the fundamental constants associated with the lattice and defined to be radius of the smallest sphere centered in the origin containing i linearly independent lattice vectors

$$\lambda_i = \lambda_i(\Lambda) = \inf\{r : \dim(\mathrm{span}(\Lambda \cap B_m(\mathbf{0}, r))) \geqslant i\} \tag{2.13}$$

where $B_m(\mathbf{0}, r)$ is the m dimensional open ball of radius r around the origin $\mathbf{0}$. There always exist linearly independent lattice vectors achieving the successive minima, i.e. there exist linearly independent $\mathbf{x}_1, \ldots, \mathbf{x}_n \in \Lambda$ such that $\|\mathbf{x}_i\| = \lambda_i$. Therefore, successive minima can also be defined as

$$\lambda_i = \lambda_i(\Lambda) = \min\left\{r : \dim\left(\operatorname{span}\left(\Lambda \cap \overline{B}_m(\mathbf{0}, r)\right)\right) \geqslant i\right\} \tag{2.14}$$

where $\overline{B}_m(\mathbf{0}, r)$ is the m dimensional closed ball of radius r around the origin $\mathbf{0}$.

Related to the determinant and the first minimum of the lattice, the Hermite's constant is defined as follows

$$\gamma_n = \max \frac{\lambda_1(\Lambda)^2}{\det(\Lambda)^{2/n}} \tag{2.15}$$

when Λ ranges over all n dimensional lattices. The Hermite's constant of dimension n has the following bounds

$$\frac{n}{2\pi e} + \frac{\log(\pi n)}{2\pi e} + o(1) \leqslant \gamma_n \leqslant \frac{1.744n}{2\pi e}(1 + o(1)). \tag{2.16}$$

Furthermore, $\gamma_2 = 4/3$ and $\gamma_n \leqslant (2/3)n$ for all $n \geqslant 2$.

2.4. Important Theorems

Theorem 2.1. *Let B be a lattice basis and B^* be the corresponding Gram-Schmidt orthogonalization. Then*

$$\lambda_1 \geqslant \min_{i=1}^{n} \|\mathbf{b}_i^*\| > 0. \tag{2.17}$$

The theorem proves a lower bound on the first successive minimum. The proof can be carried out by evaluating the expression $\langle \mathbf{Bx}, \mathbf{b}_i^* \rangle$ where $\mathbf{x} \in \mathbb{Z}^n$ by letting i be the largest index such that $x_i \neq 0$ and showing $|\langle \mathbf{Bx}, \mathbf{b}_i^* \rangle| \geqslant \|\mathbf{b}_i^*\|^2$ by using the fact that $|\langle \mathbf{Bx}, \mathbf{b}_i^* \rangle| \leqslant \|\mathbf{Bx}\| \, \|\mathbf{b}_i^*\|$.

Theorem 2.2. *Let Λ be a lattice of rank n. There exist linearly independent lattice vectors $\mathbf{v}_i$ for $i = 1, \ldots, n$ such that*

$$\|\mathbf{v}_i\| = \lambda_i. \tag{2.18}$$

In order to show that there exist a lattice vector achieving the first minimum, one can use the closed ball $\overline{B}_m(\mathbf{0}, 2\lambda_1)$. Since the closed ball is a compact set one can

extract a convergent subsequence of lattice vectors with their limit equals to $\mathbf{w}$ such that $\|\mathbf{w}\| = \lambda_1$ and then one can prove that $\mathbf{w}$ is a lattice vector belonging to the sequence. By generalizing this approach, the theorem can be proved for all successive minima.

Theorem 2.3. *(Blichfeldt Theorem) Let Λ be a lattice and $S \subseteq span(\Lambda)$ be a measurable set. If $vol(S) > \det(\Lambda)$ then there exist distinct $\mathbf{z}_1, \mathbf{z}_2 \in S$ such that $\mathbf{z}_1 - \mathbf{z}_2 \in \Lambda$.*

It is possible to prove the theorem by using the translated parallelepipeds $P_{\mathbf{x}}(\mathrm{B}) = \mathbf{x} + P(\mathrm{B})$ for all $\mathbf{x} \in \Lambda$. If one lets $\mathrm{S}_{\mathbf{x}} = \mathrm{S} \cap P_{\mathbf{x}}(\mathrm{B})$ and $\mathrm{S}'_{\mathbf{x}} = \mathrm{S}_{\mathbf{x}} - \mathbf{x}$, using the assumption that $\mathrm{vol}(\mathrm{S}) > \det(\Lambda)$, it can be seen $\mathrm{S}'_{\mathbf{x}}$ are not pairwise disjoint. Therefore, there exist $\mathbf{z} \in \mathrm{S}'_{\mathbf{x}} \cap \mathrm{S}'_{\mathbf{y}}$ with $\mathbf{z}_1 = \mathbf{z} + \mathbf{x} \in \mathrm{S}_{\mathbf{x}}$ and $\mathbf{z}_2 = \mathbf{z} + \mathbf{y} \in \mathrm{S}_{\mathbf{y}}$. Consequently, $\mathbf{z}_1 - \mathbf{z}_2 = \mathbf{x} - \mathbf{y} \in \Lambda$.

Theorem 2.4. *(Minkowski's Convex Body Theorem) Let Λ be a lattice of rank n and $S \subset span(\Lambda)$ be a convex set symmetric about the origin. If $vol(S) > 2^n \det(\Lambda)$ then there exist a nonzero lattice vector $\mathbf{v} \in S \cap \Lambda - \{\mathbf{0}\}$.*

This theorem is a corollary to the Blichfeldt theorem. For the set S symmetric about the origin with $\mathrm{vol}(\mathrm{S}) > 2^n \det(\Lambda)$, one can define another set $\mathrm{S}' = \{\mathbf{x} : 2\mathbf{x} \in \mathrm{S}\}$ with $\mathrm{vol}(\mathrm{S}') > \det(\Lambda)$. At this point one can find distinct vectors such that $\mathbf{z}_1, \mathbf{z}_2 \in$ S' . Due to symmetricity of S, $-2\mathbf{z}_2 \in$ S . By convexity of S and the Blichfeldt Theorem, one can state that nonzero $(2\mathbf{z}_1 - 2\mathbf{z}_2)/2 = \mathbf{z}_1 - \mathbf{z}_2$ is in both S and Λ.

Theorem 2.5. *(Minkowski's First Theorem) Let Λ be a lattice of rank n. The first minimum satisfies the following inequality*

$$\lambda_1 \leqslant \sqrt{n} \det(\Lambda)^{1/n}. \tag{2.19}$$

In order to obtain the inequality, the convex body theorem can be applied on a n dimensional sphere S of radius $\sqrt{n}\det(\Lambda)^{1/n}$ in $\mathrm{span}(\Lambda)$ centered at the origin. It is obvious that $\mathrm{vol}(\mathrm{S}) > 2^n \det(\Lambda)$ since the sphere contains n dimensional hypercube with the edges of length $2^n \det(\Lambda)^{1/n}$ (or more accurately, n dimensional sphere of radius r has the volume $\pi^{n/2}\Gamma(n/2+1)\, r^n$). Therefore, by convex body theorem, the sphere contains a nonzero lattice vector.

Theorem 2.6. *(Minkowski's Second Theorem) Let Λ be a lattice of rank n. The successive minima satisfy the following inequality*

$$\left(\prod_{i=1}^{n} \lambda_i\right)^{1/n} \leqslant \sqrt{n} \det(\Lambda)^{1/n}. \tag{2.20}$$

To prove the theorem, one can define a transformation using the Gram-Schmidt orthogonalized vectors $\mathrm{T}\left(\sum c_i \mathbf{x}_i^*\right) = \sum \lambda_i c_i \mathbf{x}_i^*$ where $\|\mathbf{x}_i\| = \lambda_i$ for all i. Using T, n dimensional unit sphere S in span (Λ) can be transformed to the symmetric convex body $\mathrm{T}(\mathrm{S})$. Assuming that the inequality in the theorem is not correct gives $\mathrm{vol}\left(\mathrm{T}(\mathrm{S})\right) > 2^n \det(\Lambda)$. Hence $\mathrm{T}(\mathrm{S})$ contains a nonzero lattice vector $\mathbf{y}$. Letting k be the largest index i such that $c_i \neq 0$ in $\mathbf{y} = \sum \lambda_i c_i \mathbf{x}_i^* = \mathrm{T}\left(\sum c_i \mathbf{x}_i^*\right) = \mathrm{T}(\mathbf{x})$ and k' be the smallest index such that $\lambda_{k'} = \lambda_k$, one can see that $\mathbf{y}$ is linearly independent from $\mathbf{x}_1, \ldots, \mathbf{x}_{k'-1}$ and $\|\mathbf{y}\| \leqslant \lambda_k \|\mathbf{x}\| < \lambda_k$. Consequently, $\mathbf{x}_1, \ldots, \mathbf{x}_{k'-1}, \mathbf{y}$ form k' linearly independent vectors of length $< \lambda_{k'} = \lambda_k$ contradicting the definition of k'th successive minimum. Therefore the assumption is wrong and the inequality in theorem holds.

Minkowski's theorems have stronger (or generalized) forms. To mention them briefly, for any lattice of rank , the following statements hold.

- $\lambda_1 \leqslant \sqrt{\gamma_n} \det(\Lambda)^{1/n}$
- $\left(\prod_{i=1}^{n} \lambda_i\right)^{1/n} \leqslant \sqrt{\gamma_n} \det(\Lambda)^{1/n}$

where γ_n is the Hermite's constant.

2.5. Referential

For more information on lattices, complete proofs and generalized or stronger versions of the theorems, reader might refer to the texts on the geometry of numbers, such as (Gruber and Lekkerkerker 1987, Siegel 1989, Lagarias 1995, Cassels 1997, Silverman 2006, Coppel 2006).

CHAPTER 3

LATTICE PROBLEMS

The purpose of this chapter is to give brief information on the lattice related problems of interest. The problems discussed here are the shortest vector and the closest vector problem in lattices. In addition, the complexity and the algorithmic results related to mainly these problems are mentioned.

3.1. SVP and CVP

There are many problems related to the lattices. Finding the shortest lattice vector and finding the closest lattice vector to a given target point are two such problems which only have exponential time algorithms to solve them in an exact manner.

Definition 3.1. (Shortest Vector Problem - SVP) Given a lattice Λ, find a nonzero lattice vector $\mathbf{x} \in \Lambda - \{\mathbf{0}\}$ such that $\|\mathbf{x}\| \leqslant \|\mathbf{y}\|$ for all $\mathbf{y} \in \Lambda - \{\mathbf{0}\}$.

Definition 3.2. (Closest Vector Problem - CVP) Given a lattice Λ and a target vector $\mathbf{t}$, find a lattice vector $\mathbf{x} \in \Lambda$ such that $\|\mathbf{x} - \mathbf{t}\| \leqslant \|\mathbf{y} - \mathbf{t}\|$ for all $\mathbf{y} \in \Lambda$.

In order to study the problems in depth, different versions of the problems which capture different algorithmic tasks are defined. The search problems capture the task of finding a particular lattice vector which is nonzero and the shortest (in SVP) or the closest vector to the given point (in CVP). Whereas, the optimization problems consist of finding the minimum of the norms of all lattice vectors (in SVP) or the minimum value of the distance between the lattice and the target point (in CVP). Finally, the decision problems are based on deciding whether there exists a lattice vector with norm smaller than or equal to a given positive rational value or whether there exists a lattice vector whose distance to the target point is less than or equal to the given positive rational value. The search problems are more difficult than the other versions since they include finding the specific vectors, whereas the decision problems are the easiest. For this reason, all the hardness results hold for decision problems.

Due to the hardness of these problems, approximate versions, which return solutions within some specified factor from the optimal, are defined below (with a little change in the representation).

Definition 3.3. (Approximate SVP - SVP$_\gamma$) Given a lattice basis $\mathbf{B} \in \mathbb{Z}^{m\times n}$, find a nonzero lattice vector $\mathbf{Bx}$ where $\mathbf{x} \in \mathbb{Z}^n - \{\mathbf{0}\}$ such that $\|\mathbf{Bx}\| \leqslant \gamma\|\mathbf{By}\|$ for all $\mathbf{y} \in \mathbb{Z}^n - \{\mathbf{0}\}$.

Definition 3.4. (Approximate CVP - CVP$_\gamma$) Given a lattice basis $\mathbf{B} \in \mathbb{Z}^{m\times n}$ and a target vector $\mathbf{t} \in \mathbb{Z}^m$, find a lattice vector $\mathbf{Bx}$ where $\mathbf{x} \in \mathbb{Z}^n$ such that $\|\mathbf{Bx}-\mathbf{t}\| \leqslant \gamma\|\mathbf{By}-\mathbf{t}\|$ for all $\mathbf{y} \in \mathbb{Z}^n$.

Note that the above problems are defined on integer lattices. This mainly stems from the fact that from a computational perspective the representation of real numbers are limited to a subset of rational numbers and every rational lattice has an equivalent integer lattice representation.

The approximation factor γ is generally defined as a function of the lattice dimension (or rank) although it can be a function of any parameter related to the lattice. The best known polynomial time algorithms achieve approximation factors exponential in the rank of the lattice.

Definition 3.5. (GapSVP$_\gamma$) Given an instance $(\mathbf{B}, r)$ where lattice basis $\mathbf{B} \in \mathbb{Z}^{m\times n}$ and rational number $r \in \mathbb{Q}$. For γ is a function of the rank,

- $(\mathbf{B}, r)$ is a YES instance if $\|\mathbf{Bz}\| \leqslant r$ for some $\mathbf{z} \in \mathbb{Z}^n - \{\mathbf{0}\}$.
- $(\mathbf{B}, r)$ is a NO instance if $\|\mathbf{Bz}\| > \gamma r$ for all $\mathbf{z} \in \mathbb{Z}^n - \{\mathbf{0}\}$.

Definition 3.6. (GapCVP$_\gamma$) Given an instance $(\mathbf{B}, \mathbf{t}, r)$ where lattice basis $\mathbf{B} \in \mathbb{Z}^{m\times n}$, target vector $\mathbf{t} \in \mathbb{Z}^m$ and rational number $r \in \mathbb{Q}$. For γ is a function of the rank,

- $(\mathbf{B}, \mathbf{t}, r)$ is a YES instance if $\|\mathbf{Bz}-\mathbf{t}\| \leqslant r$ for some $\mathbf{z} \in \mathbb{Z}^n$.
- $(\mathbf{B}, \mathbf{t}, r)$ is a NO instance if $\|\mathbf{Bz}-\mathbf{t}\| > \gamma r$ for all $\mathbf{z} \in \mathbb{Z}^n$.

It is possible to define r to be a real number because the real r can always be replaced with a proper rational approximation. For example, in l_2 norm a real r can always be replaced by a rational number in $\left[r, \sqrt{r^2+1}\right)$.

Note that, when $\gamma = 1$, SVP_γ and CVP_γ are equivalent to SVP and CVP, and GapSVP_γ and GapCVP_γ are equivalent to the decision problems associated with SVP and CVP respectively. Furthermore, if one has an access to an algorithm which solves SVP_γ (CVP_γ) then one can easily solve GapSVP_γ (GapCVP_γ) using this algorithm. On the other hand, if one has an access to a decision oracle solving GapSVP_γ (GapCVP_γ), one can solve SVP_γ (CVP_γ).

Another important remark is that, even if one manages to find a shortest (or a closest) lattice vector, there is no known polynomial time proof (certificate) which shows it is indeed a shortest (or closest) lattice vector.

3.2. Summary of Complexity Results

There have been several researches on the complexity of lattice problems. Here, the research regarding SVP and CVP is summarized. The topics of interest are the hardness results, which, in a sense, provide lower bounds in order to efficiently approximate the mentioned problems and the non-hardness results, which presents the factors in which the related problems do not seem to be hard. The results for l_∞ norm are mentioned separately due to the similarities between the earlier findings on the problems with respect to this norm. Furthermore, some studies on the relations between SVP and CVP are also referred.

For good surveys on the complexity of lattice problems reader may refer to (Cai 1999, 2000, Regev 2007).

3.2.1. SVP Hardness

Ajtai (1996) presents significant worst-case to average-case connection (reduction) for some special versions of SVP, which states that if there is no algorithm to approximate (decisional) SVP within some polynomial factor for any lattice, then (search) SVP is hard to solve exactly when the lattice is chosen randomly according to a certain distribution. This type of connection currently does not exist in any other problem in NP and is believed to be outside P. Later, Ajtai (1998) also proves the NP-hardness of SVP (with respect to the l_2 norm) under randomized reductions, being a breakthrough initi-

ating several studies on SVP and other lattice problems. In the same paper, Ajtai shows that approximating SVP to within factors $1+2^{-n^{\varepsilon}}$, for some absolute constant $\varepsilon > 0$, with respect to the l_2 norm is NP-hard for randomized reductions and the corresponding decision problem is NP-complete for randomized reductions.

Cai and Nerurkar (1997, 1999) improve the results of Ajtai by improving the worst-case to average-case connection and by demonstrating the NP-hardness of approximating SVP to within a factor $1+n^{-\varepsilon}$, for $\varepsilon > 0$, under randomized reductions and with respect to the l_p norms where $1 \leqslant p < \infty$.

Micciancio (1998, 2001b) further improves the hardness results and shows that in any l_p norm SVP is NP-hard to approximate to within any constant factor less than $2^{1/p}$ (in particular $\sqrt{2}$) under randomized reductions. In the same paper, Micciancio also shows that a proper NP-hardness result (hardness under deterministic many-one reductions) can be obtained under a certain reasonable number theoretic conjecture.

Kumar and Sivakumar (1999) show that the problem of deciding whether there exist a lattice vector shorter than a given rational is NP-hard under randomized reductions. The result holds even the lattice has exactly zero or one such vector.

Khot (2003) presents a new hardness of approximation result for SVP under the assumption NP $\not\subseteq$ ZPP. The result is to within a factor of $p^{1-\varepsilon}$, for $\varepsilon > 0$, with respect to l_p norm for large enough constant $p = p(\varepsilon)$. Furthermore, in (Khot 2005), the author improves the results in (Micciancio 2001b, Khot 2003) by proving that there is no polynomial time algorithm approximating SVP in l_p, norm where $1 < p < \infty$ to within any constant factor under the assumption NP $\not\subseteq$ RP and that there is no polynomial time algorithm approximating SVP in l_p, $1 < p < \infty$, norm with approximation ratio $2^{\log^{0.5-\varepsilon} n}$, $\varepsilon > 0$ is an arbitrarily small constant, under the stronger assumption that NP $\not\subseteq$ RTIME $\left(2^{\text{poly}(\log n)}\right)$. Although, Khot's proof does not work for l_1 norm, Regev and Rosen (2006) impliy that it also hold for $p = 1$.

Haviv and Regev (2007) improve the approximation factor in (Khot 2005) to an almost-polynomial level, $2^{\log^{1-\varepsilon} n}$ for any $\varepsilon > 0$ under l_p norm where $1 \leqslant p < \infty$. Furthermore, the authors also show that under the (stronger) assumption NP $\not\subseteq$ RSUBEXP = $\bigcap_{\delta>0}$ RTIME $\left(2^{n^{\delta}}\right)$ the hardness factor is $n^{c/\log\log n}$ for some constant $c > 0$.

On the other hand, there has been some research on lattice problems addressing particular quantum-related concepts. Regev (2004c) presents the first explicit connec-

tion between quantum computation and lattice problems, particularly on SVP. Later, Aharonov and Regev (2003) give the first non-trivial upper bound on the quantum complexity of a lattice problem by showing that coGapSVP (a Gap version of SVP which has been known to be in AM $\cap$ coNP but not known to be in NP or MA) lies in QMA, the quantum analogue of NP. Also, Regev (2005) gives a quantum reduction from worst case lattice problems SVP and SIVP (shortest independent vectors problem) to a certain learning problem and proposes a public key cryptosystem whose security is based on the hardness of the learning problem and so on the worst-case quantum hardness of SVP and SIVP.

3.2.2. CVP Hardness

van Emde Boas (1981) is the first one to establish the NP-hardness of CVP, and Kannan (1987b, 1983) gives a more natural proof of this result.

Arora et al. (1997) prove that approximating CVP within a constant factor in any l_p norm is NP-hard. They also show that approximating CVP to within a factor of $2^{\log^{0.5-\varepsilon} n}$ for $\varepsilon > 0$, is hard, unless NP is in quasi-polynomial deterministic time DTIME($n^{\text{poly}(\log n)}$).

Dinur et al. (1998) show that CVP in lattice is NP-hard to approximate to within almost-polynomial factors, more accurately any factor up to $2^{\log^{1-\varepsilon} n}$ where $\varepsilon = (\log\log n)^{-a}$ for any constant $a < 1/2$, improving (Arora, et al. 1997). Furthermore, together with Raz, they prove that CVP is NP-hard to approximate to within $n^{c/\log\log n}$ for some constant $c > 0$ with respect to the l_p norms where $1 \leqslant p < \infty$ (Dinur, et al. 2003).

Cai (2001) establishes a worst-case to average-case connection for CVP. Thus, an evidence of average-case hardness of CVP is provided.

3.2.3. SVP and CVP Non-hardness

Lagarias at al. (1990) state that approximating SVP or CVP to within a factor of cn, for an appropriate constant c, cannot be NP-hard, unless NP = coNP. Furthermore, combining (Lagarias, et al. 1990, Hastad 1988, Banaszczyk 1993) shows that SVP and

CVP within n is in NP $\cap$ coNP. Therefore, a factor n NP-hardness would imply that NP = coNP.

Goldreich and Goldwasser (2000) prove that approximating SVP or CVP, under the smart Cook reductions, to within a factor of $\sqrt{n}$, more specifically $\sqrt{n/\log n}$, is unlikely to be NP-hard since the problem lies in NP $\cap$ coAM and an NP-hardness with such factors would imply coNP $\subseteq$ AM.

Cai (1998) simplifies the proof of the argument in (Lagarias, et al. 1990), and states that finding a $n^{1/4}$-unique shortest vector is not NP-hard under polynomial time many-one reductions, unless the polynomial time hierarchy collapses. In addition, in (Cai and Nerurkar 2000), it is shown that the arguments in (Goldreich and Goldwasser 2000) is also valid even under general Cook reductions.

Aharonov and Regev (2005) improve the factor n non-hardness result and show that approximating SVP or CVP to within a factor of $c\sqrt{n}$, for some $c > 0$, lies in NP $\cap$ coNP, meaning that achieving a $c\sqrt{n}$ NP-hardness would imply that NP = coNP. Furthermore, following the results of Goldreich and Goldwasser (2000), the authors remark that approximating SVP or CVP to within a factor of $c\sqrt{n/\log n}$ is likely to be NP $\cap$ coNP.

3.2.4. Results in l_∞ Norm

It is notable that SVP and CVP behave similarly with respect to the l_∞ norm and they seem to be more difficult. NP-hardness of SVP and CVP with respect to the l_∞ norm is established by van Emde Boas (1981).

Arora et al. (1997) state that approximating SVP in l_∞ norm to within a factor of $2^{\log^{0.5-\varepsilon} n}$ for $\varepsilon > 0$, is hard, unless NP is in quasi-polynomial deterministic time, DTIME($n^{\text{poly}(\log n)}$). In the same paper, the result is proved to hold for CVP in any l_p norm, in particular CVP in l_∞ norm. Furthermore, it is shown that improving the factor to $\sqrt{n}$ would imply the hardness of SVP in l_2 norm.

Dinur (2000) shows that SVP and CVP under norm l_∞ is NP-hard to approximate to within an almost-polynomial factor $n^{1/\log\log n}$, improving the results of Arora et al. (1997), and, in (Dinur 2002), further generalizes the result by proving that SVP and CVP with respect to the l_∞ norm is NP-hard to approximate to within almost-polynomial

factors $n^{c/\log\log n}$ for some constant $c > 0$.

Goldreich and Goldwasser (2000) give non-hardness result on SVP and CVP with respect to l_∞ norm showing that $n/\log n$ factor NP-hardness for SVP or CVP in l_∞ norm would imply the collapse of polynomial time hierarchy.

3.2.5. SVP-CVP Reductions

Some of the important results regarding the reductions between SVP and CVP can be briefly and chronologically listed as follows.

Kannan (1987a) provides a link relating approximating SVP to approximating CVP. Approximating CVP to a factor of $n^{3/2} f(n)^2$ is polynomial time Turing reducible (Cook reducible) to approximating SVP to a factor of $f(n)$ for any nondecreasing function $f(n)$. Also, the author proves that approximating CVP to within a factor of $\sqrt{n/2}$ is polynomial time Cook reducible to the decision version of SVP (Kannan 1987b).

Henk (1997) shows with respect to certain class of norms there exist a polynomial time Turing, more specifically Cook, reduction from SVP to CVP.

Goldreich et al. (1999) show that approximating SVP is not harder than approximating CVP by presenting a direct reduction from SVP to CVP which preserves the factor of approximation and lattice dimension. Specifically, given an oracle which solves $f(n)$-approximate CVP, one can solve $f(n)$-approximate SVP in polynomial time.

Ajtai et al. (2002) present a $2^{O(n)}$ time Turing reduction from approximate CVP to SVP.

3.3. Summary of Algorithmic Results

As already mentioned, there is no polynomial time algorithm solving SVP or CVP. Furthermore, there is no known such algorithm to approximate these problems to within a polynomial factor. The best polynomial time algorithms achieve slightly subexponential factors. In this section, main algorithmic results for (approximately) solving SVP and CVP are summarized.

3.3.1. Approximating SVP

Lenstra et al. (1982) describe and analyze a lattice basis reduction algorithm by improving the Lenstra's algorithm in (Lenstra 1981). The algorithm is came to be known as LLL (Lenstra Lenstra Lovasz) algorithm and it has several variants such as (Schönhage 1984, Schnorr 1988, Schnorr and Euchner 1994, Storjohann 1996, Koy and Schnorr 2001a,b, Akhavi 2002, Koy and Schnorr 2002, Schnorr 2006b, Nguyen and Stehlé 2007). Roughly, LLL algorithm can be used to approximate SVP to within a factor of $2^{(n-1)/2}$ performing $O\left(n^4 \log B\right)$ number of arithmetic operations on integers of binary length $O(n \log B)$, where B is the squared norm bound of the input basis vectors. On the other hand, it is also possible to achieve the provable approximation factor up to $\alpha^{(n-1)/2}$, where $\alpha \approx 4/3$, by choosing the parameters for basis reduction differently, while preserving the practicality. It is true that $\alpha^{(n-1)/2} \lessapprox 1.153^n$ for $\alpha \approx 4/3$. The factor of approximation can be decreased to for lattices with high density where $\lambda_1^2 \approx \gamma_n \det(\Lambda)^{2/n}$. Furthermore, for random lattices in (Nguyen and Stehlé 2006), the value of α can be decreased to ≈ 1.08 on the average for the LLL in (Lenstra, et al. 1982), and to ≈ 1.05 on the average for the LLL with deep insertions in (Schnorr and Euchner 1994). See (Nguyen and Stehlé 2006, Schnorr 2006a).

Schnorr (1987) defines a family of reduction algorithms from LLL to HKZ (Hermite Korkine Zolotarev) reduction, named BKZ (Block Korkine Zolotarev), whose performance depends on , the blocksize, parameter. The author improves and uses the (HKZ) reduction algorithm proposed by Kannan (1983) in order to obtain polynomial time algorithms to approximate SVP. Schnorr's blockwise lattice basis reduction algorithm runs in $O\left(n^2\left(k^{k/2+o(k)}+n^2\right)\log B\right)$ arithmetic operations performed on $O(n \log B)$ bit integers. The algorithm approximates SVP to within a factor of $\left(6k^2\right)^{n/(2k)}$ where assuming the unproven bound $\gamma_k \leqslant k/6$ for $k \geqslant 24$, one obtains a factor of $(k/3)^{n/k}$. Gama et al. (2006) show the factor to be $(\beta_k/\delta)^{(n-1)/(2k)}$ where β_k is a constant related to BKZ reduced bases and $\delta \approx 1$. As stated by Schnorr (2006a), for $k = 24$, this factor is $< 1.165^{(n-1)/2}$ which, under the heuristic in (Schnorr 2003), improves to $\leqslant \alpha^{(n-1)/2}$ where $\alpha \approx 1.034$. Furthermore, for the proper choice of the blocksize, $k = O(\log n / \log\log n)$, one can achieve a $2^{O\left(n(\log\log n)^2/\log n\right)}$ approximation factor.

Kumar and Sivakumar (2001) show a $2^{O(n/\varepsilon)}$ time algorithm which approximates SVP to within a factor $n^{3+\varepsilon}$ for arbitrary $\varepsilon > 0$.

Ajtai et al. (2001), based on (Kumar and Sivakumar 2001), present a probabilistic sieve algorithm which finds the shortest vectors. Using this algorithm in Schnorr's BKZ basis reduction algorithm, it is possible to achieve $2^{O(n\log\log n/\log n)}$ approximation to SVP in randomized polynomial time for $k = O(\log n)$. Furthermore, as stated by Schnorr (2001), assuming the unproven bound $\gamma_k \leqslant k/6$ for $k \geqslant 24$ and combining the sieve algorithm with the reduction algorithm in (Koy 2004) yields a factor of $(k/6)^{n/k}$ in $O\left(n^2 2^{O(k)} + n^4\right)$ time storing $2^{O(k)}$ lattice vectors.

Schnorr (2003) describes an algorithm which approximates SVP to within $(k/6)^{n/2k}$ in $O\left(n^3 (k/6)^{k/4} + n^4\right)$ time by iterating random sampling of short lattice vectors. In addition, it is stated that the algorithm can be improved using some particular additional heuristics at the expense of increased space complexity. Inspired by Ajtai et al. (2001), the author also presents a sieve algorithm to approximate SVP to within a factor of $k^{(3n)/(4k)}$ in $O\left(n^3 2^{0.835k} + n^4\right)$ time storing $2^{0.835k} + O(n)$ lattice vectors. Furthermore, comparisons with the other approximation algorithms are performed, and some refinements are proposed on the results in (Schnorr 2001).

Koy (2004) introduces a blockwise lattice basis reduction for approximating SVP. The algorithm achieves a proven factor of $\left(\alpha\gamma_k^2\right)^{n/k-1}$ where γ_k is the Hermite's constant and $\alpha \approx 4/3$ (for $k = 48$ approximation factor is proven to be $\approx 1.075^{(n-1)/2}$). The algorithm runs in $O\left(n^3 m \log_{1/\delta} 2\right) + n^3 k^{O(k)}$ arithmetic steps where $\delta \approx 1$ and the runtime is also expressed as $O\left(n^3 k^{k/2+o(k)} + n^4\right)$ time. Assuming the unproven bound $\gamma_k \leqslant k/6$ for $k \geqslant 24$, the factor becomes $(k/6)^{n/k}$. In addition, under the heuristic of Schnorr (2003), the factor can be improved to $\approx (1.034)^{(n-1)/2}$. Moreover, it is possible to further decrease the approximation factor to $\approx (1.025)^{(n-1)/2}$ for $k = 80$ by replacing the HKZ reduction by the random sampling reduction in (Schnorr 2003) under the same heuristic. The details of the Koy's algorithm can also be found in (Schnorr 2006a).

Blömer and Naewe (2007) obtain single-exponential time $1+\varepsilon$ approximation algorithm for SVP requiring $\left((2+1/\varepsilon)^n b\right)^{O(1)}$ arithmetic operations where $\varepsilon > 0$ is arbitrary.

Gama and Nguyen (2008a) present a deterministic and a probabilistic algorithm which perform $\gamma_k^{(n-k)/(k-1)}$ approximation to SVP. The deterministic algorithm is an im-

provement of the algorithms in (Schnorr 1987, Gama, et al. 2006) and achieves a factor of $2^{O\left(n(\log\log n)^2/\log n\right)}$ for $k = O(\log n/\log\log n)$ in polynomial time. Whereas the probabilistic algorithm, using the randomized algorithm of Ajtai et al. (2001) within blocks of size k, solves SVP to within a factor of $2^{O(n(\log\log n)/\log n)}$ for $k = O(\log n/\log\log n)$ in polynomial time.

As a last remark on the approximation algorithms, Gama and Nguyen (2008b) notes that it seems reasonable to assume that current algorithms, namely LLL, BKZ and their particular variants, should achieve in a reasonable time an approximation factor $\leqslant 1.01^n$ on the average, and $\leqslant 1.02^n$ in the worst case based on the extensive experiments. This is an important assessment of the optimistic behaviour of the lattice basis reduction algorithms, i.e. the lattice basis reduction algorithms generally behave much better than their proven worst case theoretical bounds in practice.

3.3.2. Solving SVP

Kannan (1983, 1987b) uses LLL in order to solve SVP. The super-exponential algorithm of Kannan performs $n^{n+o(n)}s^{O(1)}$ arithmetic operations, where s is the length of the input. The numbers produced in the execution of the algorithm are of binary length $O\left(n^2(\log B + \log n)\right)$. Helfrich (1985) improves the running time of the Kannan's algorithm for SVP to $n^{n/2+o(n)}s^{O(1)}$ arithmetic operations. In addition, Hanrot and Stehlé (2007, 2008) further analyze the Kannan's algorithm and show its running time complexity to be $n^{n/2e+o(n)}s^{O(1)}$ arithmetic operations.

Ajtai et al. (2001) present a sieve algorithm for solving SVP. The algorithm, which has $2^{O(n)}$ space bound, can be used to solve SVP (with high probability) in a randomized $2^{O(n)}$ time improving $2^{O(n\log n)}$ time bound of Kannan's algorithm for the problem. In fact, in $2^{O(n)}$ time the algorithm finds all approximate vectors with approximation factor greater than or equal to 1. The practicality of the algorithm is shown by Nguyen and Vidick (2008).

3.3.3. Approximating CVP

Babai (1986) uses LLL to obtain a reduced lattice basis in order to approximate

CVP. The author presents two polynomial time algorithms based on two different heuristics. The first algorithm, rounding off algorithm, approximates CVP to within a factor of $1+2n\,(9/2)^{n/2}$ whereas the second algorithm, namely the nearest plane algorithm, achieves a $2^{n/2}$ approximation factor, which can be improved to $2\,(4/3)^{n/2}$.

Schnorr (1996) presents a BKZ version of the nearest plane algorithm. The algorithm can be used to approximate CVP to within a factor of $\sqrt{n}\gamma_k^{n/(k-1)}$ in $k^{O(k)}$ time. Therefore, for the properly chosen blocksize parameter, the approximation factor becomes $2^{O\left(n(\log\log n)^2/\log n\right)}$ in polynomial time. This factor can also be achieved using the algorithms in (Gama, et al. 2006, Gama and Nguyen 2008a).

Ajtai et al. (2001) obtain an algorithm to solve approximate CVP to within a factor of $\sqrt{n/2}$ in randomized $2^{O(n)}$ time (combining a $2^{O(n)}$ time algorithm for SVP and the polynomial time Turing reduction from approximate CVP to SVP given in (Kannan 1987a)). In addition, it is possible to approximate CVP to within a factor of $2^{O(n\log\log n/\log n)}$ using their randomized polynomial time SVP algorithm and the Kannan's link presented in (Kannan 1987a). Moreover, in (Ajtai, et al. 2002), they present a randomized $2^{O\left(1+\varepsilon^{-1}\right)n}$, for arbitrary $\varepsilon>0$, algorithm which performs $(1+\varepsilon)$ approximation to CVP, using the SVP algorithm from (Ajtai, et al. 2001) as the subroutine. Hence, they improve the existing time bound of $O(n!)$ for CVP achieved by the deterministic algorithm in (Blömer 2000).

Blömer and Naewe (2007) approximates CVP, in addition to SVP, to within a factor $1+\varepsilon$ in probabilistic single-exponential time. The algorithm has the complexity bound $\left((2+1/\varepsilon)^n s\right)^{O(1)}$ for arbitrary $\varepsilon>0$.

3.3.4. Solving CVP

Kannan (1983, 1987b) also presents an algorithm to solve CVP which uses his algorithm to solve SVP introduced in the same paper. As in the case of SVP, the algorithm performs $n^{n+o(n)}s^{O(1)}$ number of arithmetic operations which is decreased to $n^{n/2+o(n)}s^{O(1)}$ by Hanrot and Stehlé (2007). The numbers produced during the execution are rational numbers with numerators and denominators having binary length $O\left(n^2\,(s+\log n)\right)$.

Blömer (2000) introduces a new technique based on dual HKZ bases. Using

this technique, CVP is solved in $n!s^{O(1)}$ time achieving an exponential improvement ($n^n/n! \approx e^n$) over Helfrich's improvement over Kannan's algorithm, (Helfrich 1985, Kannan 1987b).

Klein (2000) proposes an algorithm which runs in $n^{k^2+O(1)}$ time and finds the closest vector to a given target vector under the condition that the distance of the target vector from the lattice is at most k times the length of the shortest Gram-Schmidt vector. The result is a generalization of the argument in (Furst and Kannan 1989) where k is taken to be $1/2$.

Agrell et al. (2002) propose an algorithm to solve CVP. They state that the algorithm is faster than Kannan's algorithm (Kannan 1987b) by at least a factor of $(2n/\pi e)^{n/2}$.

3.4. Basis Reduction and Other Lattice Problems

It is important to note the algorithms for solving (approximate) SVP / CVP, do more that just solving them. These algorithms also produce a reduced basis. Aside from SVP and CVP, finding a "good" lattice basis is one of other important problem related to the lattices. This problem is generally referred to as basis reduction problem. There is no unique definition of good basis. One could search for a basis where the maximum of the lengths of its vectors is minimized or one could look for a basis where the product of the vector lengths is minimized. Many important problems related to the lattice basis reduction are discussed in (Micciancio and Goldwasser 2002).

For more information on (other) lattice-related problems, such as unique SVP (a special version of SVP), Hermite-SVP (another variation of SVP), closest vector problem with preprocessing (CVPP), shortest independent vectors problem (SIVP), shortest basis problem (SBP) and covering radius problem (CRP) etc, reader may refer to the resources, including but not limited to, (Ajtai 1996, Blömer and Seifert 1999, Cai 1999, 2000, Micciancio and Goldwasser 2002, Guruswami, et al. 2005, Regev 2005, Haviv and Regev 2006, Chen and Meng 2006, Blömer and Naewe 2007, Micciancio 2008). In addition, Regev and Rosen (2006), Peikert (2007) present interesting relations between lattice problems in l_2 norm and the corresponding problems in other l_p norms.

3.5. Referential

For more information on the complexity, or cryptocomplexity, related issues, one may refer to textbooks such as (Garey and Johnson 1990, Micciancio and Goldwasser 2002, Rothe 2005).

CHAPTER 4

LENSTRA LENSTRA LOVASZ REDUCTION

LLL algorithm which is first proposed by Lenstra et al. (1982) is the first algorithm to approximately solve SVP. The algorithm's runtime is polynomial and the achieved SVP approximation factor is exponential in the lattice dimension, more accurately $\alpha^{n/2}$ where $\alpha \approx 4/3$. In this chapter, we shall discuss the main aspects of the LLL algorithm and try to emphasize its importance.

4.1. LLL Reduced Basis

The projection operation is defined as, given $\mathbf{x} \in \mathbb{R}^m$,

$$\pi_i(\mathbf{x}) = \sum_{j=i}^{n} \frac{\langle \mathbf{x}, \mathbf{b}_j \rangle}{\langle \mathbf{b}_j, \mathbf{b}_j \rangle} \mathbf{b}_j^* \tag{4.1}$$

where $\mathbf{b}_1^*, \ldots, \mathbf{b}_n^*$ are the Gram-Schmidt ortogonalized vectors. The resulting vector $\pi_i(\mathbf{x})$ is the component of the vector $\mathbf{x}$ orthogonal to span$(\mathbf{b}_1, ..., \mathbf{b}_{i-1})$.

Definition 4.1. (δLLL Reduced Basis) A basis $\mathbf{B} = [\mathbf{b}_1 \ldots \mathbf{b}_n] \in \mathbb{R}^{m \times n}$ is δLLL reduced if

- $|\mu_{i,j}| \leqslant \frac{1}{2}$ for all $i > j$ where $\mu_{i,j} = \frac{\langle \mathbf{b}_i, \mathbf{b}_j^* \rangle}{\langle \mathbf{b}_j^*, \mathbf{b}_j^* \rangle}$ are the Gram-Schmidt coefficients,
- $\delta \|\pi_i(\mathbf{b}_i)\|^2 \leqslant \|\pi_i(\mathbf{b}_{i+1})\|^2$ where $\pi_i(\mathbf{x})$ is the component of the vector $\mathbf{x}$ orthogonal to span$(\mathbf{b}_1, ..., \mathbf{b}_{i-1})$ and $1/4 < \delta < 1$ is a real number.

Note that $\pi_i(\mathbf{b}_i) = \mathbf{b}_i^*$ and $\pi_i(\mathbf{b}_{i+1}) = \mathbf{b}_{i+1}^* + \mu_{i+1,i}\mathbf{b}_i^*$. Furthermore, since the vectors $\mathbf{b}_1, \ldots, \mathbf{b}_n$ are orthogonal we have $\left\|\mathbf{b}_{i+1}^* + \mu_{i+1,i}\mathbf{b}_i^*\right\|^2 = \left\|\mathbf{b}_{i+1}^*\right\|^2 + |\mu_{i+1,i}|^2 \|\mathbf{b}_i^*\|^2$. Therefore, if a basis is δLLL reduced then one can write that

$$\delta \|\mathbf{b}_i^*\|^2 \leqslant \left\|\mathbf{b}_{i+1}^*\right\|^2 + |\mu_{i+1,i}|^2 \|\mathbf{b}_i^*\|^2. \tag{4.2}$$

Arranging the inequality one obtains

$$\left(\delta - \frac{1}{4}\right) \|\mathbf{b}_i^*\|^2 \leqslant \left\|\mathbf{b}_{i+1}^*\right\|^2. \tag{4.3}$$

By induction on the orthogonalized vectors it is obvious that

$$\left(\delta-\frac{1}{4}\right)^{i-1}\|\mathbf{b}_1^*\|^2 \leqslant \|\mathbf{b}_i^*\|^2. \tag{4.4}$$

For $1/4<\delta<1$, one can write $(\delta-1/4)^{i-1} \geqslant (\delta-1/4)^{n-1}$, and since $\|\mathbf{b}_1^*\| = \|\mathbf{b}_1\|$, it follows that

$$\left(\delta-\frac{1}{4}\right)^{n-1}\|\mathbf{b}_1\|^2 \leqslant \|\mathbf{b}_i^*\|^2. \tag{4.5}$$

Combining this with the fact that $0<\min_{i=1}^{n}\|\mathbf{b}_i^*\| \leqslant \lambda_1$, one gets

$$\left(\delta-\frac{1}{4}\right)^{(n-1)/2}\|\mathbf{b}_1\| \leqslant \lambda_1. \tag{4.6}$$

Consequently,

$$\|\mathbf{b}_1\| \leqslant \left(\frac{4}{4\delta-1}\right)^{(n-1)/2}\lambda_1. \tag{4.7}$$

Thus, one approximates SVP to within a factor of $(4/(4\delta-1))^{(n-1)/2}$ by δLLL reducing the basis where $1/4<\delta<1$. Note that for $\delta=3/4$ the approximation factor becomes $2^{(n-1)/2}$ as in (Lenstra, et al. 1982).

In the light of the above discussion, let $\alpha=(\delta-1/4)^{-1}$ then for all LLL reduced bases one can write

- $\|\mathbf{b}_1\|^2 \leqslant \alpha^{n-1}\lambda_1^2$ and $\|\mathbf{b}_1\|^2 \leqslant \alpha^{i-1}\|\mathbf{b}_i^*\|^2$ for all $i=1,\ldots,n$,
- $\|\mathbf{b}_1\|^2 \leqslant \alpha^{(n-1)/2}\det(L(\mathrm{B}))^{2/n}$ and $\|\mathbf{b}_n^*\|^2 \geqslant \alpha^{-(n-1)/2}\det(L(\mathrm{B}))^{2/n}$,

using the fact that $\|\mathbf{b}_i^*\|^2 \leqslant \alpha\left\|\mathbf{b}_{i+1}^*\right\|^2$. Furthermore, generalizing the discussion in (Lenstra, et al. 1982), it is possible to obtain

$$\alpha^{1-i} \leqslant \|\mathbf{b}_i\|^2/\lambda_i^2 \leqslant \alpha^{n-1} \tag{4.8}$$

for all LLL reduced bases, where $i=1,\ldots,n$.

4.2. LLL Basis Reduction Algorithm

The LLL lattice basis reduction algorithm can be seen in Figure 4.1. It is important to make some remarks on the main points of the algorithm which are crucial in order to grasp the underlying idea.

Remarks on the reduction step (or the size reduction) of the LLL algorithm shown in Figure 4.1 can be given as follows:

- The reduction is also called the size reduction step and transforms the basis into an equivalent basis because only elementary (column) operations are performed.
- The Gram-Schmidt orthogonalized vectors associated to the basis before and after the reduction step remains the same, because if $i > j$ the operation $\mathbf{b}_i \leftarrow \mathbf{b}_i - a\mathbf{b}_j$ does not change the Gram-Schmidt orthogonalization.
- The ith iteration of the outer loop guarantees that $\left|\mu_{i,j}\right| \leqslant 1/2$ for $i > j$, because

$$\left|\mu_{i,j}\right| = \left|\frac{\left\langle \mathbf{b}_i - c_{i,j}\mathbf{b}_j, \mathbf{b}_j^* \right\rangle}{\left\langle \mathbf{b}_j^*, \mathbf{b}_j^* \right\rangle}\right| = \left|\frac{\left\langle \mathbf{b}_i, \mathbf{b}_j^* \right\rangle}{\left\langle \mathbf{b}_j^*, \mathbf{b}_j^* \right\rangle} - \left[\frac{\left\langle \mathbf{b}_i, \mathbf{b}_j^* \right\rangle}{\left\langle \mathbf{b}_j^*, \mathbf{b}_j^* \right\rangle}\right] \frac{\left\langle \mathbf{b}_j, \mathbf{b}_j^* \right\rangle}{\left\langle \mathbf{b}_j^*, \mathbf{b}_j^* \right\rangle}\right| \leqslant \frac{1}{2} \tag{4.9}$$

  which follows from the fact that $\left\langle \mathbf{b}_j, \mathbf{b}_j^* \right\rangle = \left\langle \mathbf{b}_j^*, \mathbf{b}_j^* \right\rangle$.
- It is crucial that the inner loop goes from $i-1$ down to 1.

Furthermore, there are some points to be made on the swap step of the LLL algorithm:

- The swap step is performed to satisfy the condition $\delta \left\|\pi_i(\mathbf{b}_i)\right\|^2 \leqslant \left\|\pi_i(\mathbf{b}_{i+1})\right\|^2$ for all i.
- There might be several pairs violating this condition and which pair is swapped does not matter in terms of correctness of the algorithm. In fact, it is possible to swap several disjoint pairs at the same time.
- After the swap the previous steps of the algorithm need to be redone because the basis might not satisfy the condition $\left|\mu_{i,j}\right| \leqslant 1/2$ anymore.

Let $B \geqslant \left\|\mathbf{b}_i\right\|^2 \in \mathbb{R}$ for $1 \leqslant i \leqslant n$, then the LLL algorithm has the following properties.

- SVP approximation factor is $\alpha^{(n-1)/2}$ where $\alpha = (\delta - 1/4)^{-1}$ and $1/4 < \delta < 1$ ($\alpha = 2$ for $\delta = 3/4$ and $\alpha \approx 4/3$ for $\delta \approx 1$).
- Algorithm performs $O\left(n^3 m \log B\right)$ number of arithmetic and $O\left(n^5 m \log^3 B\right)$ bit operations.
- Integers on which the arithmetic operations are performed have binary length $O(n \log B)$.

input : Lattice Basis $\mathbf{b}_1, \ldots, \mathbf{b}_n \in \mathbb{Z}^m$ and δ

output: A δLLL reduced basis for the lattice $L(\mathrm{B})$

(start):

compute $\mathbf{b}_1^*, \ldots, \mathbf{b}_n^*$

(reduction):

for $i = 2$ to n do

    for $j = i - 1$ to 1 do

        $\mathbf{b}_i \leftarrow \mathbf{b}_i - \left\lceil c_{i,j} \right\rfloor \mathbf{b}_j$ where $c_{i,j} = \left[\left\langle \mathbf{b}_i, \mathbf{b}_j^* \right\rangle / \left\langle \mathbf{b}_j^*, \mathbf{b}_j^* \right\rangle \right]$

    end

end

(swap):

if $\delta \|\pi_i(\mathbf{b}_i)\|^2 > \|\pi_i(\mathbf{b}_{i+1})\|^2$ is true for some i then

    $\mathbf{b}_i \leftrightarrow \mathbf{b}_{i+1}$

    goto (start)

end

output $\mathbf{b}_1, \ldots, \mathbf{b}_n$

Figure 4.1. LLL Lattice Basis Reduction Algorithm.

4.3. The Overview of Analysis

This section provides outline on the proof of correctness and the running time analysis of the LLL algorithm. For the complete proof and analysis, one might refer to (Lenstra, et al. 1982, Micciancio and Goldwasser 2002, Regev 2004a).

4.3.1. Note on the Correctness

One can easily show that if the algorithm terminates it satisfies the conditions

- $\left|\mu_{i,j}\right| \leqslant \frac{1}{2}$ for all $i > j$
- $\delta \left\|\pi_i(\mathbf{b}_i)\right\|^2 \leqslant \left\|\pi_i(\mathbf{b}_{i+1})\right\|^2$

using the remarks given in the previous section which hint on the correctness of the algorithm.

4.3.2. Note on the Running Time

The running time analysis of the algorithm consists of two steps. The first step is bounding the number of iterations performed by the algorithm, which is exactly the number of swap operations performed by the algorithm. The second step contains bounding the running time of each iteration and showing that the size of numbers produced in the execution of algorithm is also bounded.

In order to bound the number of iterations, one can associate a positive integer to the lattice basis, let $D = \prod_{i=1}^{n} \det(\Lambda_i)^2$ where $\Lambda_i = L(\mathbf{b}_1, \ldots, \mathbf{b}_i)$ is the sublattice generated by the first i basis vectors. It can be shown that the value of D changes only when the swap occurs. Let D_{k+1} be the new value, assuming the vectors $\mathbf{b}_i$ and $\mathbf{b}_{i+1}$ are swapped at the kth iteration, it is straightforward to note that D_k decreases by at least by a factor of δ,

$$\frac{D_{k+1}}{D_k} = \frac{\det(\Lambda_i')^2}{\det(\Lambda_i)^2} = \frac{\left\|\pi_i(\mathbf{b}_{i+1})\right\|^2}{\left\|\pi_i(\mathbf{b}_i)\right\|^2} < \delta. \tag{4.10}$$

It follows that $1 \leqslant D_k < \delta^k D_0$ and the number of iterations can be bounded by

$$k < \log_{1/\delta} D_0 \leqslant \frac{1}{\log(1/\delta)} n(n-1) \log\left(\max_{i=1}^{n} \|\mathbf{b}_i\|\right) \tag{4.11}$$

which follows from the fact that $\|\mathbf{b}_i^*\| \leqslant \|\mathbf{b}_i\|$ and that $D_0 \leqslant \left(\max_{i=1}^n \|\mathbf{b}_i\|\right)^{n(n-1)}$. Since $\log\left(\max_{i=1}^n \|\mathbf{b}_i\|\right)$ is polynomial in the input size, it can be seen that the number of iterations is polynomial in the input size for constant $\delta < 1$.

In order to bound the running time of each iteration, one should note that the number of arithmetic operations performed at each iteration is polynomial. Therefore, to show a polynomial bound on the running time it is enough to show that the numbers arising in each iteration can be represented using polynomial number of bits (leading to the fact that the actual running time which is the number of the bit operations performed is also polynomial). Since the numbers are rational, one should bound both the precision and the magnitude.

To bound the precisions of numbers, one can write

$$\langle \mathbf{b}_i - \mathbf{b}_i^*, \mathbf{b}_k \rangle = \left\langle \sum_{j=1}^{i-1} v_{i,j}\mathbf{b}_j, \mathbf{b}_k \right\rangle = \langle \mathbf{b}_i, \mathbf{b}_k \rangle = \sum_{j=1}^{i-1} v_{i,j} \left\langle \mathbf{b}_j, \mathbf{b}_k \right\rangle \tag{4.12}$$

for $k < i$ and $v_{i,j} \in \mathbb{R}$. For $k = 1, \ldots i-1$, there is a system of $i-1$ linear equations in $i-1$ variables

$$\mathbf{b}_i^{\mathrm{T}} \mathbf{B}_{i-1} = \mathbf{v}_i^{\mathrm{T}} \mathbf{B}_{i-1}^{\mathrm{T}} \mathbf{B}_{i-1} \tag{4.13}$$

where $\mathbf{B}_k = [\mathbf{b}_1 \ldots \mathbf{b}_k]$ and $\mathbf{v}_i = [v_{i,1} \ldots v_{i,i-1}]^{\mathrm{T}}$. It is possible to interpret $\mathbf{v}_i$ as the solution vector and $\mathbf{B}_{i-1}^{\mathrm{T}} \mathbf{B}_{i-1}$ as the coefficient matrix. Letting $d_{i-1} = \det(\Lambda_{i-1})^2 = \det \mathbf{B}_{i-1}^{\mathrm{T}} \mathbf{B}_{i-1} \in \mathbb{Z}$ when the Cramer's rule is applied, it can be seen that $d_{i-1}\mathbf{v}_i$ is an integer vector and therefore $d_{i-1}\mathbf{b}_i^* = d_{i-1}\mathbf{b}_i + \sum_{j=1}^{n} d_{i-1} v_{i,j} \mathbf{b}_j$ is also an integer vector. This shows that all denominators occurring in $\mathbf{b}_i^*$ are factors of d_{i-1}. In addition,

$$\mu_{i,j} = \frac{\left\langle \mathbf{b}_i, \mathbf{b}_j^* \right\rangle}{\left\langle \mathbf{b}_j^*, \mathbf{b}_j^* \right\rangle} = \frac{d_{j-1} \left\langle \mathbf{b}_i, \mathbf{b}_j^* \right\rangle}{d_{j-1} \left\| \mathbf{b}_j^* \right\|^2} = \frac{\left\langle \mathbf{b}_i, d_{j-1}\mathbf{b}_j^* \right\rangle}{d_j} \tag{4.14}$$

so denominators of $\mu_{i,j}$ divide d_j. Thus, denominators of all rationals divide $D = \prod_{i=1}^n d_i$. Since $\log D$ is polynomial in the input size, all denominators occurring in the computation can be represented using polynomially many bits.

To bound the magnitude of the numbers it is sufficient to show that

$$\|\mathbf{b}_i^*\|^2 = d_i \prod_{j=1}^{i-1} \left(\left\| \mathbf{b}_j^* \right\|^2 \right)^{-1} \leqslant d_i \prod_{j=1}^{i-2} d_j^2 \leqslant D^2 \tag{4.15}$$

since $\left\| \mathbf{b}_j^* \right\|^{-1} \leqslant d_{j-1}$ where $d_0 = 1$, and that

$$\|\mathbf{b}_i\|^2 = \|\mathbf{b}_i^*\|^2 + \sum_{j=1}^{i-1} \mu_{i,j}^2 \left\| \mathbf{b}_j^* \right\|^2 \leqslant D^2 + (n/4) D^2 \leqslant nD^2 \tag{4.16}$$

where $\left|\mu_{i,j}\right| \leqslant 1/2$. Concluding that all the quantities occurring in the computation can be represented with polynomially many bits.

4.3.3. Note on the Approximation Factor

The best approximation factor the algorithm achieves in polynomial time can be obtained by setting $\delta = (1/4) + (3/4)^{n/(n-1)}$. For such value, δ is closer to 1 than any constant and the approximation factor is $(4/3)^{n/2}$. In addition, for sufficiently large n, the algorithm still has a polynomial running time. See (Micciancio and Goldwasser 2002) for further details.

Experimentally, the approximation factor is improved on the average by Nguyen and Stehlé (2006). The approximation factor is approximately less than or equal to $\alpha^{n-1/2} \lessapprox 1.155^n$ for $\alpha \approx 4/3$, which improves to $\lessapprox 1.075^n$ for the lattices of high density where $\lambda_1^2 \approx \gamma_n \det(\Lambda)^{2/n}$. Nguyen and Stehlé (2006) shows that on random lattices $\alpha = 1.02^4 \approx 1.08$ f for the floating variant of LLL, proposed by Schnorr and Euchner (1994), and $\alpha \approx 1.05$ for LLL with deep insertions, which is also proposed by Schnorr and Euchner (1994), on the average. Also see (Schnorr 2006a, 2007).

4.4. LLL Basis Reduction Algorithm Rewritten

Figure 4.1 is more suited for the theoretical analysis. In order to emphasize the algorithmic details and aid the rest of the discussion, the LLL algorithm can be rewritten as in Figure 4.2. Note that the-always-frowned-upon goto statement has gone.

It is possible to make some adjustments in the representation of elements in Figure 4.2. Let $\mathbf{B} = \mathbf{QR}$ be the QR-factorization of the basis matrix where

$$\mathbf{Q} = \left[\begin{array}{cccc} \mathbf{b}_1^* / \|\mathbf{b}_1^*\| & \mathbf{b}_2^* / \|\mathbf{b}_2^*\| & \dots & \mathbf{b}_n^* / \|\mathbf{b}_n^*\| \end{array} \right] \in \mathbb{R}^{m \times n} \tag{4.17}$$

is an orthogonal matrix ($\mathbf{Q}^{\mathrm{T}}\mathbf{Q} = \mathbf{I}$), and

$$\mathbf{R} = \left[\begin{array}{cccc} \mathbf{r}_1 & \mathbf{r}_2 & \dots & \mathbf{r}_n \end{array} \right] \in \mathbb{R}^{n \times n} \tag{4.18}$$

is an upper triangular matrix. The matrix $\mathbf{R}$ is the orthogonalization of $\mathbf{B}$ and is referred as the geometric normal form of $\mathbf{B}$, denoted by $\mathbf{R} = \mathrm{GNF}(\mathbf{B})$. The matrix $\mathbf{Q}$ is said

input : Lattice Basis $\mathbf{b}_1, \ldots, \mathbf{b}_n \in \mathbb{Z}^m$ and δ

output: A δLLL reduced basis for the lattice $L(\mathrm{B})$

(initialization):

$l \leftarrow 1$ (l is generally referred as stage)

while $l \leqslant n$ do

compute the Gram-Schmidt coefficients $\mu_{l,1}, \ldots, \mu_{l,l-1}$ and $\left\|\mathbf{b}_l^*\right\|^2$

(size reduce $\mathbf{b}_l$ against $\mathbf{b}_{l-1}, \ldots, \mathbf{b}_1$):

for $i = l-1$ to 1 do

$\mathbf{b}_l \leftarrow \mathbf{b}_l - \left\lfloor \mu_{l,i} \right\rceil \mathbf{b}_i$

(update the Gram-Schmidt coefficients $\mu_{l,i}, \ldots, \mu_{l,1}$):

for $j = 1$ to i do

$\mu_{l,j} = \mu_{l,j} - \left\lfloor \mu_{l,i} \right\rceil \mu_{i,j}$

end

end

(swap):

if $l \neq 1$ and $\delta \left\|\mathbf{b}_{l-1}^*\right\|^2 > \mu_{l,l-1}^2 \left\|\mathbf{b}_{l-1}^*\right\|^2 + \left\|\mathbf{b}_l^*\right\|^2$ then

$\mathbf{b}_{l-1} \leftrightarrow \mathbf{b}_l$

$l \leftarrow l-1$

else

$l \leftarrow l+1$

end

end

output $\mathbf{b}_1, \ldots, \mathbf{b}_n$

Figure 4.2. LLL Lattice Basis Reduction Algorithm (2).

to be isometric, since $\langle \mathbf{Qx}, \mathbf{Qy} \rangle = \langle \mathbf{x}, \mathbf{y} \rangle$ and can be extended to an orthogonal matrix $\mathbf{Q}' \in \mathbb{R}^{m \times m}$. Two bases $\mathbf{B}$ and $\mathbf{B}'$ are isometric if and only if $\mathrm{GNF}(\mathbf{B}) = \mathrm{GNF}(\mathbf{B}')$. The geometric normal form and the LLL reduction are preserved under isometric transforms $\mathbf{Q}$, i.e. $\mathrm{GNF}(\mathbf{B}) = \mathrm{GNF}(\mathbf{QB})$. Furthermore, due to the fact that $\langle \mathbf{b}_i, \mathbf{b}_j \rangle = \langle \mathbf{r}_i, \mathbf{r}_j \rangle$, $\mu_{j,i} = r_{i,j}/r_{i,i}$ and $\|\mathbf{b}_i^*\| = r_{i,i}$, the LLL algorithm can be equivalently written as the algorithm in Figure 4.3.

Algorithm in Figure 4.3 is notable because it works with the QR-factorization. This gives the implication that the Gram-Schmidt orthogonalization can be replaced due to the fact that there are other methods, such as Householder reflections and Givens rotations to compute the QR-factorization, to find the orthogonalized vectors of a given basis. Therefore, one can substitute the Gram-Schmidt orthogonalization if it is beneficial.

Another important point is that, at the lth stage, LLL performs local LLL reductions in the swap step on the submatrices

$$\begin{bmatrix} r_{l-1,l-1} & r_{l-1,l} \\ 0 & r_{l,l} \end{bmatrix}, \tag{4.19}$$

and so computes the shortest vector in the 2 dimensional sublattices.

4.5. LLL Variants

The LLL algorithm has many variants. Most remarkable modifications are the usage of floating point arithmetic numbers, the deep insertions and carrying out the local reductions on (sub)segments. However, many of such variants have improvable results. In general, they depend on heuristics for practicality. On the other hand, there also other variants which are obtained by relaxing or tightening the conditions posed by the classical LLL algorithm in order to produce at least equivalently good reduced bases.

4.5.1. LLL with Floating Point Arithmetic

A problem with the LLL algorithm is that it uses arithmetic operations on large integers with $O(n \log B)$ bits. In fact, it is somewhat possible to avoid the overhead of the large integer arithmetic by performing computation using approximate real numbers and floating point arithmetic. However, this approach comes with stability problems and

input : Lattice Basis $\mathbf{b}_1, \ldots, \mathbf{b}_n \in \mathbb{Z}^m$ and δ

output: A δLLL reduced basis for the lattice $L(\mathrm{B})$

(initialization):

$l \leftarrow 1$ (l is the stage)

while $l \leqslant n$ do

(compute $\mathbf{r}_l = \left[r_{1,l}, \ldots, r_{l,l}, 0, \ldots, 0\right]^T = \left\|\mathbf{b}_l^*\right\| \left[\mu_{l,1}, \ldots, \mu_{l,l}, 0, \ldots, 0\right]^T$):

for $i = 1$ to $l-1$ do

$r_{i,l} \leftarrow \left(\langle \mathbf{b}_i, \mathbf{b}_l \rangle - \sum_{k=1}^{i-1} r_{k,i} r_{k,l}\right) / r_{i,i}$

end

$r_{l,l} \leftarrow \left| \|\mathbf{b}_l\|^2 - \sum_{k=1}^{l-1} r_{k,l}^2 \right|^{1/2}$

(size reduce $\mathbf{b}_l$ and $\mathbf{r}_l$):

for $i = l-1$ to 1 do

$\mathbf{b}_l \leftarrow \mathbf{b}_l - \left[\mu_{l,i}\right] \mathbf{b}_i$

$\mathbf{r}_l \leftarrow \mathbf{r}_l - \left[r_{i,l}/r_{i,i}\right] \mathbf{r}_i$

end

(swap):

if $l \neq 1$ and $\delta r_{l-1,l-1}^2 > r_{l-1,l}^2 + r_{l,l}^2$ then

$\mathbf{b}_{l-1} \leftrightarrow \mathbf{b}_l$

$l \leftarrow l-1$

else

$l \leftarrow l+1$

end

end

output $\mathbf{b}_1, \ldots, \mathbf{b}_n$

Figure 4.3. LLL Lattice Basis Reduction Algorithm (3).

in order to preserve sufficient level of accuracy additional measures should be taken. One such measure is the generalization of the size reduction condition $|\mu_{i,j}| \leqslant \eta$ where, in general, $\eta \in \left[1/2, \sqrt{\delta}\right)$. This stems from the fact that the original size reduction condition ($\eta = 1/2$) can not be achieved when floating point arithmetic numbers are used. Furthermore, one should also note that the basis should be kept in exact representation since the errors occurring in the computation of the basis changes the whole lattice, therefore cannot be corrected. However, it is possible to recover the other errors by using a correct basis.

Schnorr (1988) proposes a modification to LLL applying the method of self-correction to approximate the inverse of a given matrix. The algorithm has provably negligible floating point errors. The proposed reduction algorithms works on $O(n + \log B)$ bit integers requiring $O\left(n^3 m \log B\right)$ number of arithmetic steps. Furthermore, when combined with the semi-reduction of Schönhage (1984), the required number of arithmetic steps decreases to $O\left(n^{2.5} m \log B\right)$. On the other hand, Nguyen and Stehlé (2007) remarks that this algorithm is mostly of theoretical interest and is not implemented widely, which might possibly be explained by the following reasons: it is not clear which floating point arithmetic model is used, the algorithm is not easy to describe, and the hidden complexity constants are rather large. More precisely, the required precision of floating point numbers seems to be higher than $12n + 7\log_2 B$.

Schnorr and Euchner (1994) introduce a practical floating point variant of LLL, named LLLFP, which makes use of floating point arithmetic. The floating point arithmetic is used to calculate only the Gram-Schmidt coefficients because otherwise the algorithm would be unstable due to occurring floating point errors and error propagation. Furthermore, LLLFP introduces different means to correct the errors occurring in the computation due to the floating point approximations. The produced lattice vectors are considerably shorter than those of LLL and the algorithm is 10% faster than the practical versions of the algorithm in (Schnorr 1988). The authors state that LLLFP has a good stability since it is empirically tested up to dimension 150 using 300 bit integers. They remark that the algorithm's stability is good even for single precision floating point arithmetic for very large numbers and the use of double precision numbers decreases the number of swaps significantly, and to offset small floating point errors one has to use δ values greater than $1/4$, i.e. $\delta \geqslant 1/2$. In addition, as noted by Backes and Wetzel (2001),

both the run time and the stability of the algorithm strongly depend on the precision of the approximations used. High precision approximations cause major loss in efficiency whereas the small precision causes the algorithm not to terminate due to (accumulated) floating point errors. Therefore, the algorithm still lacks of efficiency in particular for large lattice bases or bases with large entries.

Backes and Wetzel (2001) present a heuristic for the LLLFP algorithm proposed in (Schnorr and Euchner 1994). The key idea is to work with the original lattice basis for the exact representation so that the problems caused by high precision approximations can be prevented. The new heuristic allows dynamic adaptation of the floating point precision thus decreases the run time of the reduction process considerably. The authors state that the new heuristic is more efficient in reducing large problem instances, allows a major decrease of the run time and extends the applicability of the Schnorr-Euchner algorithm such that problem instances that the state-of-the-art method fails to reduce can be solved. Furthermore, the same authors, in (Backes and Wetzel 2002), present empirical results on the practical performance of the LLL algorithms, and introduces new heuristics to improve the LLL variant proposed in (Schnorr and Euchner 1994) together with test data on these methods. The heuristics developed for the Schnorr-Euchner LLL strive to speed up the computation, by eliminating the unnecessary operations or doing operations on shorter operands, and to achieve better reduction results, possibly causing a decrease in the running time. After performing several tests using the new heuristics, Backes and Wetzel propose general heuristics for different classes of lattices in order to determine which variant of the reduction algorithm, for varied parameter choices, yields the most efficient reduction strategy for reducing a particular problem instance.

Nguyen and Stehlé (2007) present a new and natural floating point variant of the LLL algorithm, namely LL algorithm, which provably outputs an LLL reduced basis in polynomial time, $O\left(n^4 m\,(n+\log B)\log B\right)$, using only $n\log_2 3$ bit precision which is independent of $\log B$. The algorithm is called LL because it is is the first LLL algorithm whose (actual) running time grows only quadratically with respect to $n\log_2 3$. The LL algorithm keep a sufficiently good floating point approximation to the Gram-Schmidt orthogonalization instead of keeping exact integers and the rational LLL is simulated by the chosen floating point precision linear in dimension while improving the accuracy of the usual Gram-Schmidt computations by a systematic use of the Gram matrix. Fur-

thermore, size reduction is replaced with a more stable version adapted from Babai's nearest plane algorithm. It performs more operations but better suited for floating point arithmetic.

4.5.2. Deep Insertions

Schnorr and Euchner (1994) also introduce another LLL variant called LLL with deep insertions or shortly LLLDEEP besides the floating point variant of the LLL algorithm. LLLDEEP replaces and extends the swap step of the classical LLL algorithm by the deep insertion step to possibly insert the vector $\mathbf{b}_l$, called the vector at the lth stage, to some minimally chosen position $i < l$. This decreases the norm of the ith vector $\mathbf{b}_i$ by at least a factor of $\sqrt{\delta}$. It is implemented using the floating point arithmetic model in LLLFP, however exact artihmetic model can also be used. The floating point implementation of LLLDEEP yields equivalently good results but it is possible that, in the worst case, it runs in super-polynomial time. Fortunately, it is possible to guarantee the polynomial runtime by a slight modification. As a side note, Schnorr and Euchner also show how to extend LLLFP and LLLDEEP algorithms so that the input basis can be a generator system of the lattice.

4.5.3. Segment LLL

(Koy and Schnorr 2001a,b, 2002, Schnorr 2006b) present and analyze several variants of the LLL algorithm based on heuristic arguments.

LLLH is a floating point variant of LLL which uses Householder reflections to compute the Gram-Schmidt coefficients due to the better accuracy of the method when compared to Gram-Schmidt for orthogonalization. The algorithm requires $O\left(n^2 m \log_{1/\delta} M\right)$ arithmetic steps using $n + \log_2 M_0$ bit integers using t bit floating point arithmetic numbers, where $M_0 = \max_{i=1}^{n} \|\mathbf{b}_i\|$ is the length of the basis, $M = \max_{i=1}^{n} \left\{\det(\Lambda_i)^2, 2^n\right\}$ is the volume of the basis and t is a parameter related to the underlying floating point arithmetic model. Furthermore, assuming $M_0 = 2^{O(n)}$ and $m = O(n)$ yields a running time of $O\left(n^4 m\right)$ arithmetic steps or $n^{6+O(1)}$ bit operations using $O(n)$ bit integers.

SLLL0 is a basic segment LLL reduction algorithm, which is faster than LLLH by a factor of n in the number of arithmetic steps but uses longer integer and floating point arithmetic numbers. It is based on the concept of segment reduced basis, which is a weaker form of the notion of the LLL reduced basis. The basis is partitioned into q segments of size k, where $n = qk$, and the algorithm performs of the local LLL exchanges in two consecutive segments using coordinates of dimension $2k$. For the number of segments $k = \Theta(\sqrt{n})$ and $M_1 = \max_{1 \leqslant i \leqslant j \leqslant n} \left\{ \|\mathbf{b}_i^*\| / \left\|\mathbf{b}_j^*\right\| \right\}$, SLLL0 runs in $O\left(nm \log_{1/\delta} M\right)$ arithmetic steps using $2n + \log_2\left(M_0 M_1^2\right)$ bit integers, and assuming $M_0 = 2^{O(n)}$, SLLL0 performs $O\left(n^3 m\right)$ arithmetic steps using $O\left(n^2\right)$ bit integers.

SLLL is the segment LLL reduction algorithm which improves the longer integer size of SLLL0. The algorithm is firstly introduced in (Koy and Schnorr 2001a), implemented in (Koy and Schnorr 2001b) using floating point orthogonalization, and lastly revised in (Schnorr 2006b). SLLL requires $O\left(nm \log_2 n \log_{1/\delta} M\right)$ arithmetic operations on the integers of binary length $2n + \log_2 M_0$. Furthermore, for $M_0 = 2^{O(n)}$ and $m = O(n)$, runtime is $O\left(n^4 \log n\right)$ arithmetic steps or $n^{5+O(1)}$ bit operations.

SLLL+ is SLLL reduction via iterated subsegments where an iterative structure of segments of levels $\sigma = 1, \ldots, s$ is used and segments of level σ are partitioned into segments of level $\sigma - 1$. So the concept of SLLL basis is extended by relaxation of some particular conditions. SLLL+ is mentioned and refined through (Koy and Schnorr 2001a, 2002, Schnorr 2006b). It saves a factor n in the number of arithmetic steps required when compared to SLLL, however it also requires more precision bits. The algorithm performs $O\left(n^2 m + n \log_2 n \log_{1/\delta} M\right)$ arithmetic operations on the integers of binary length $O(\log_2 M_0 M) = O(n + \log_2 M_0)$. In particular, for $M_0 = 2^{O(n)}$ and $m = O(n)$, it runs in $O\left(n^3 \log n\right)$ arithmetic steps or $n^{5+O(1)}$ bit operations.

Strong versions of SLLL and SLLL+ are discussed in (Koy and Schnorr 2002). Basically, they are referred as algorithms which have similar runtimes with their normal counterparts but produce bases satisfying stronger conditions.

Note that SLLL and SLLL+ are firstly proposed in (Koy and Schnorr 2001a) over an integer arithmetic model. Therefore, they do not enforce the usage of floating point arithmetic numbers.

4.5.4. Other Variants

Kaltofen (1983) shows that the LLL reduction can be performed $O\left(n^6 \log^2 B + n^5 \log^3 B\right)$ arithmetic steps by analyzing a modified version of the algorithm which also performs better in practice.

Schönhage (1984) improves LLL by proposing a semi-reduction algorithm which works in segments to reduce the basis. The algorithm speeds up LLL by a factor of n. However the quality of the reduced basis is not as good. The runtime of the algorithm is $O\left(n^2 m \log B\right)$ arithmetic operations performed on $O\left(n \log B\right)$ bit integers and the achieved factor is 2^n.

Pohst (1987) modifies the LLL algorithm in such a way that the input vectors can also be linearly dependent, i.e. a generating system. The output of the algorithm consists of a basis of the lattice generated by the input vectors and non-trivial combinations of 0 by the input vectors if they are linearly dependent.

Storjohann (1996) considers LLL as a matrix algorithm and replaces size reduction with modular reduction to obtain an algorithm which requires $O\left(n^2 m \log B\right)$ arithmetic operations using standard matrix multiplication. In addition, by combining the new algorithm with the semi-reduction of Schönhage (1984) and employing faster multiplication techniques, number of arithmetic operations needed to be performed drops to $O\left(n^{1.381} m \log B\right)$.

Akhavi (2002) presents two lattice basis reduction algorithms by relaxing some of the constraints of the LLL algorithm. In general, the new algorithms require fewer iterations but still produce good reduced basis. Both the algorithms share a common parameter $s = (\delta - 1/4)^{-1}$, and the algorithms achieve the same approximation factor with LLL in the worst case. The author also provides an average case analysis and performs empirical tests.

Gama et al. (2006) study some particular class of lattices, namely symplectic lattices, and obtains symplectic LLL reduction algorithm. They show that orthogonalization techniques are compatible with symplecticity to obtain an algorithm for Gram-Schmidt and a variant of LLL which is empirically shown to be (6-times) faster than the classical LLL algorithm. They also present some optimizations for further speed up.

Akhavi and Stehlé (2008) describe a randomized algorithm which, with very

high probability, computes a lattice bases satisfying properties similar to those returned by LLL. The algorithm applies a dimension reducing random projection to the basis before the LLL reduction is performed and after the reduction inverse transformation is applied to obtain the reduced basis.

4.6. Brief Notes

4.6.1. The importance of LLL

LLL plays an important role in the lattice basis reduction and in the computation of shortest and closest vectors. For more information on how LLL is used in the practice of basis reduction and exactly/approximately solving hard lattice problems refer to the resources (in the chronological order) including but no limited to (Kannan 1983, Helfrich 1985, Babai 1986, Schnorr 1987, Kannan 1987a,b, Buchmann and Kessler 1990, Aardal 1999, Schnorr 2001, 2003, 2006a, Gama, et al. 2006).

4.6.2. Generalized Lattice Basis Reduction Algorithm

Lovász and Scarf (1992) generalizes the LLL algorithm to work with polynomial time computable generalized distance function on convex bodies. The classical LLL is identical to the special case where the bodies are ellipsoids.

4.6.3. Optimal and Average Case LLL

Daudé and Vallée (1994) give an upper bound on the average number of iterations of the LLL algorithm, which is $O\left(n^2 \log n\right)$ thus independent of the basis length and $O(1)$ in fixed dimension. Whereas, Akhavi (2003) shows that the optimal LLL (LLL with $\delta = 1$) is polynomial, since the number of iterations of the algorithm is linear, in the fixed dimension. Furthermore, Nguyen and Stehlé (2006) investigate the mysteriously optimistic behaviour of lattice basis reduction algorithms by trying to model the average case of the algorithms, starting with the celebrated LLL, and giving experimental bounds on the approximation factors.

4.6.4. Parallel LLL

There are also parallelized versions of the LLL algorithm. For more information reader should refer to (Villard 1992, Roch and Villard 1992, Heckler and Thiele 1993a,b, 1998).

4.7. Referential

For more and detailed information on the contemporary LLL algorithms, one should refer to (Stehlé 2007), which presents a good in depth survey on floating point variants of the LLL algorithm while comparing them in detail, and (Schnorr 2007), which surveys almost all the LLL algorithms, their use in the lattice basis reduction and the achieved approximation factors for the shortest lattice vector problem by making helpful comments and comparisons. Furthermore, Nguyen and Stehlé (2006) provide experimental facts on the average case behaviour of the LLL algorithm, and Gama and Nguyen (2008b) provide an insight on what is achievable today with the best lattice reduction algorithms known in terms of output quality and running time, based on extensive experiments performed and shed new lights on the practical hardness of the main lattice problems.

CHAPTER 5

BLOCK KORKINE ZOLOTAREV REDUCTION

Schnorr (1987) proposes a family of polynomial time blockwise lattice basis reduction algorithms which form an hierarchy between two most popular basis reduction concepts, the LLL (Lenstra Lenstra Lovasz) basis redcution and the HKZ (Hermite Korkine Zolotarev) basis reduction. These algorithms are called BKZ (Blockwise Korkine zolotarev) basis reduction algorithms and achieve very good approximation factors for SVP. In general, the achieved polynomial time approximation factors for SVP are stated as $(1+\varepsilon)^n$ where the value of ε changes according to the algorithm in use, the chosen parameter values and in some cases the effects of the heuristics applied.

5.1. HKZ, k - and Block $2k$ - Reduced Bases

The projected lattice $\pi_i(L(\mathrm{B}))$ is defined for given $L(\mathrm{B})$ where $\mathbf{B} = [\mathbf{b}_1 \dots \mathbf{b}_n] \in \mathbb{R}^{m\times n}$ as the projection of the lattice $L(\mathrm{B})$ to span $(\mathbf{b}_i^*, \dots, \mathbf{b}_n^*)$ where $\mathbf{b}_1^*, \dots, \mathbf{b}_n^*$ are the Gram-Schmidt orthogonalized vectors. The vectors of the projected lattice are orthogonal to span $(\mathbf{B}_{i-1})$.

Definition 5.1. (HKZ Reduced Basis) A basis $\mathbf{B} = [\mathbf{b}_1 \dots \mathbf{b}_n] \in \mathbb{R}^{m\times n}$ is HKZ reduced if

- $\left|\mu_{i,j}\right| \leqslant \frac{1}{2}$ for all $i > j$ where $\mu_{i,j}$ are the Gram-Schmidt coefficients,
- $\mathbf{b}_i^*$ is a shortest nonzero vector in the projected lattice $\pi_i(L(\mathrm{B}))$.

Note that if linearly independent set of lattice vectors is HKZ reduced then they form a basis for the original lattice and, by Lagarias et al. (1990) , every HKZ reduced basis satisfies

$$\frac{4}{i+3} \leqslant \frac{\|\mathbf{b}_i\|^2}{\lambda_i^2} \leqslant \frac{i+3}{4} \tag{5.1}$$

for $i = 1, \dots, n$. Furthermore

$$\|\mathbf{b}_i^*\|^2 \leqslant \gamma_{n-i+1}^{(n-i+1)/(n-i)} \left(\prod_{j=i}^{n} \|\mathbf{b}_j^*\|^2\right)^{1/(n-i+1)} \tag{5.2}$$

also holds for any HKZ reduced basis.

Definition 5.2. (k-Reduced Basis) A basis $\mathbf{B} = [\mathbf{b}_1 \dots \mathbf{b}_n] \in \mathbb{R}^{m \times n}$ is k-reduced if

- $\left|\mu_{i,j}\right| \leqslant \frac{1}{2}$ for all $i > j$ where $\mu_{i,j}$ are the Gram-Schmidt coefficients,
- the k-blocks $\pi_i(\mathbf{b}_i), \dots, \pi_i(\mathbf{b}_{i+k-1})$, $i = 1, \dots, n-k+1$, are HKZ reduced.

Definition 5.3. ($2k$-Reduced Basis) A basis $\mathbf{B} = [\mathbf{b}_1 \dots \mathbf{b}_n] \in \mathbb{R}^{m \times n}$ is $2k$-reduced if, for $n = qk$,

- $\left|\mu_{i,j}\right| \leqslant \frac{1}{2}$ for all $i > j$ where $\mu_{i,j}$ are the Gram-Schmidt coefficients,
- the $2k$-blocks $\pi_{ik+1}(\mathbf{b}_{ik+1}), \dots, \pi_{ik+1}(\mathbf{b}_{ik+2k})$, $i = 0, \dots, q-2$, are HKZ reduced.

It follows from the definitions that every $2k$-reduced basis is block $2k$-reduced, every 2-reduced basis is LLL reduced and every n-reduced basis is HKZ reduced.

It is important to note that there is no polynomial time algorithm to HKZ or k- or block $2k$- reduce any given basis. However, the properties associated to them are essential for the construction and analysis of BKZ algorithms which achieve polynomial run times.

In order to measure the quality of the produced bases, two constants are introduced. The first one is α_k which is defined as

$$\alpha_k = \max \frac{\|\mathbf{b}_1\|^2}{\left\|\mathbf{b}_k^*\right\|^2} \tag{5.3}$$

where the maximum is taken over all HKZ reduced bases of rank k lattices. Schnorr (1987) shows that

- $\alpha_2 = 4/3$ and $\alpha_k \leqslant \alpha_{k+1}$ for all k because every HKZ reduced basis $\mathbf{b}_2, \dots, \mathbf{b}_{k+1}$ extends to a HKZ reduced basis $\mathbf{b}_1, \mathbf{b}_2, \dots, \mathbf{b}_{k+1}$ by adjoining an arbitrary vector $\mathbf{b}_1$ such that $\mathbf{b}_1$ is orthogonal to span$(\mathbf{b}_2, \dots, \mathbf{b}_{k+1})$ and $\|\mathbf{b}_1\| = \|\mathbf{b}_2\|$.
- $\alpha_k \leqslant k^{1+\ln k}$ for $k \geqslant 2$ and therefore $\lim_{k\to\infty} \alpha_k^{1/k} = 1$.

The second constant is

$$\beta_k = \max \left(\frac{\|\mathbf{b}_1^*\|^2 \dots \left\|\mathbf{b}_k^*\right\|^2}{\left\|\mathbf{b}_{k+1}^*\right\|^2 \dots \left\|\mathbf{b}_{2k}^*\right\|^2} \right)^{1/k} \tag{5.4}$$

where the maximum is taken over all HKZ reduced bases of rank $2k$ lattices. β_k is also shown to satisfy the following properties in (Schnorr 1987).

- $\beta_k \leqslant 4k^2$
- $\beta_k \leqslant \prod_{i=1}^{k} \gamma_{2k-i+1}^{2/(2k-i)}$ where in particular $\beta_1 = 4/3$.

Using the constants defined above, the following statements hold according to Schnorr (1987). For any k-reduced basis

- $\|\mathbf{b}_1\|^2 \leqslant \alpha_k^{(n-1)/(k-1)} \lambda_1^2$ provided that $k-1$ divides $n-1$,
- $\|\mathbf{b}_1\|^2 \leqslant (1+\varepsilon_k)^{n-1} \lambda_1^2$ where $\lim_{k\to\infty} \varepsilon_k = 0$, since $\alpha_k \leqslant k^{1+\ln k}$ for $k \geqslant 2$.

Furthermore, for any block $2k$-reduced basis (also for any $2k$-reduced basis), with $n = qk$, we have

- $\|\mathbf{b}_1\|^2 \leqslant \gamma_k \beta_k^{q-1} \lambda_1^2$,
- $\|\mathbf{b}_1\|^2 \leqslant \left(4k^2\right)^{n/k} \lambda_1^2$, since $\beta_k \leqslant 4k^2$ and $\gamma_k \leqslant (2/3)\, k \leqslant 4k^2$ for $k \geqslant 2$.

The bounds for k-reduced bases are generalized and improved in (Schnorr 1996) as

$$\frac{4}{i+3} \gamma_k^{-2\frac{i-1}{k-1}} \leqslant \|\mathbf{b}_i\|^2 / \lambda_i^2 \leqslant \frac{i+3}{4} \gamma_k^{2\frac{n-1}{k-1}} \tag{5.5}$$

for $i = 1, \ldots, n$.

Furthermore, the bounds on α_k and β_k are improved as follows.

- $\alpha_k \geqslant k^{\varepsilon \ln k}$ and $\beta_k \geqslant k^{\varepsilon}$ for some $\varepsilon > 0$ by Ajtai (2003), however no particular value of ε is known. Thus, the results shows that the bounds given by Schnorr, $\alpha_k \leqslant k^{1+\ln k}$ and $\beta_k \leqslant 4k^2$, are tight apart from a constant factor in the exponent.
- $k/12 \leqslant \beta_k \leqslant (1+k/2)^{2\ln 2 + 1/k}$ by Gama et al. (2006) . In particular, $\beta_k \leqslant 0.38k^{2\ln 2}$ and $\lim_{k\to\infty} \beta_k^{1/k} = 1$.
- $\alpha_k \leqslant k^{(\ln k)/2 + O(1)}$ and $\beta_k \leqslant k^{2\ln 2 + O(1/\ln k)}$ by Hanrot and Stehlé (2008).

5.2. Semi k - and Semi Block $2k$ - Reduced Bases

In order to obtain polynomial time algorithms for blockwise basis reduction, the conditions for k-reduced and block $2k$-reduced bases are relaxed and HKZ reductions are performed on only the pairwise disjoint blocks. This type of reduction is called semi reduction.

To state the definition of semi reduced bases, let

$$C_i = \prod_{j=1}^{k} \left\| b^*_{ik+j} \right\|^2 \quad \text{and} \quad D_i = \prod_{j=0}^{i-1} C_i \tag{5.6}$$

for $i = 0, \ldots, q-1$.

Definition 5.4. (Semi k-Reduced Basis) A basis $\mathbf{B} = [\mathbf{b}_1 \ldots \mathbf{b}_n] \in \mathbb{R}^{m \times n}$ for $n = qk$ is semi k-reduced if

- $\left\| \mathbf{b}^*_{ik} \right\|^2 \leqslant \alpha \left\| \mathbf{b}^*_{ik+1} \right\|^2$ for $i = 0, \ldots, q-1$,
- the k-blocks $\pi_{ik+1}(\mathbf{b}_{ik+1}), \ldots, \pi_{ik+1}(\mathbf{b}_{ik+k})$, $i = 0, \ldots, q-1$, are HKZ reduced,

where $1/4 < \delta < 1$ and $\alpha = (\delta - 1/4)^{-1}$.

Definition 5.5. (Semi Block $2k$-Reduced Basis) A basis $\mathbf{B} = [\mathbf{b}_1 \ldots \mathbf{b}_n] \in \mathbb{R}^{m \times n}$ for $n = qk$ is semi block $2k$-reduced if

- $\delta^k C_i \leqslant \beta_k^k C_{i+1}$ for $i = 1, \ldots, q-1$,
- $\left\| \mathbf{b}^*_{ik} \right\|^2 \leqslant \alpha \left\| \mathbf{b}^*_{ik+1} \right\|^2$ for $i = 1, \ldots, q-1$,
- the k-blocks $\pi_{ik+1}(\mathbf{b}_{ik+1}), \ldots, \pi_{ik+1}(\mathbf{b}_{ik+k})$, $i = 1, \ldots, q-1$, are HKZ reduced,

where $1/4 < \delta < 1$ and $\alpha = (\delta - 1/4)^{-1}$.

It follows from the definitions that every block $2k$-reduced basis is semi block $2k$-reduced and every k-reduced basis is semi k-reduced.

In general, if $\mathbf{b}_1, \ldots, \mathbf{b}_n$ is a semi k-reduced basis with $n = qk$, one can write that

$$\|\mathbf{b}_1\|^2 \leqslant \alpha^{q-1} \alpha_k^q \lambda_1^2 \tag{5.7}$$

holds. Furthermore, if the basis is semi block $2k$-reduced, then

$$\|\mathbf{b}_1\|^2 \leqslant \alpha \gamma_k \alpha_k \left(\frac{\beta_k}{\delta} \right)^{q-2} \lambda_1^2 \tag{5.8}$$

is also true.

Schnorr (1987) uses $1/\delta = 4/3$ and $\alpha = 2$ to permit proving a polynomial time bound, remarking that it is possible to replace $4/2$ by any number greater than 1, and 2 by any number greater than $4/3$. He carries out the analysis and states the above

results in the following particular forms using the specified values. If $\mathbf{b}_1, \ldots, \mathbf{b}_n$ is a semi k-reduced basis where $n = qk$, then

$$\|\mathbf{b}_1\|^2 \leqslant 2^{q-1}\alpha_k^q \lambda_1^2 \tag{5.9}$$

holds and if the basis is a semi block $2k$-reduced basis, then

$$\|\mathbf{b}_1\|^2 \leqslant 2\gamma_k \alpha_k \left(\frac{4}{3}\beta_k\right)^{q-2} \lambda_1^2 \tag{5.10}$$

which in particular yields

$$\|\mathbf{b}_1\|^2 \leqslant \left(6k^2\right)^{n/k} \lambda_1^2. \tag{5.11}$$

Gama et al. (2006) give the following inequalities concerning the approximation factor of semi block $2k$-reduction,

$$\|\mathbf{b}_1\| \lessapprox \sqrt{\gamma_k} \left(\frac{4}{3}\right)^{(3k-1)/4} \beta_k^{(n/k-2)/2} \lambda_1 \tag{5.12}$$

and

$$\|\mathbf{b}_1\| \lessapprox \sqrt{\gamma_k} \beta_k^{n/4k} \det\left(L\left(\mathrm{B}\right)\right)^{1/n}. \tag{5.13}$$

Furthermore, Schnorr (2006a, 2007) gives the following inequalities which are shown to be optimal by Ajtai (2003). Any semi block $2k$-reduced basis satisfies

$$\|\mathbf{b}_1\|^2 / \lambda_1^2 \leqslant k^{\ln k + o(\ln k)} \left(\beta_k/\delta\right)^{n/k-2} \tag{5.14}$$

where $o\left(\ln k\right)$ in the exponent can be replaced by 2 for $k \geqslant 3$, and

$$\|\mathbf{b}_1\|^2 / \det\left(L\left(\mathrm{B}\right)\right)^{2/n} \leqslant \gamma_k \left(\beta_k/\delta\right)^{(n/k-1)/2} \tag{5.15}$$

where $1/4 < \delta < 1$. Thus, it is possible to write for any semi block $2k$-reduced basis

$$\|\mathbf{b}_1\| \leqslant \left(\beta_k/\delta\right)^{(n-1)/(2k)} \lambda_1 \tag{5.16}$$

and

$$\|\mathbf{b}_1\| \leqslant \left(\beta_k/\delta\right)^{(n-1)/(4k)} \det\left(L\left(\mathrm{B}\right)\right)^{1/n} \tag{5.17}$$

where $n/k \to \infty$.

5.3. Semi k-Reduction and Semi Block $2k$-Reduction Algorithms

In order to refer to the blocks more concisely in the algorithms, the following entities shall be used.

- $S_i = \{\pi_{ik+1}(\mathbf{b}_{ik+1}), \ldots, \pi_{ik+1}(\mathbf{b}_{ik+k})\}$ are k-blocks for $i = 0, \ldots, q-1$,
- $L_i = \{\pi_{ik+1}(\mathbf{b}_{ik+1}), \ldots, \pi_{ik+1}(\mathbf{b}_{ik+2k})\}$ are $2k$-blocks for $i = 0, \ldots, q-2$.

Note that the k-blocks S_i, $i = 0, \ldots, q-1$, are disjoint, whereas the $2k$-blocks L_i, $i = 0, \ldots, q-2$, are not.

The algorithms for semi k - and semi block $2k$ - reduction are given in Figure 5.1 and Figure 5.2 respectively. It might be a good idea to apply the LLL reduction to the the input bases during the initialization steps of the algorithms. Furthermore, as in the case of LLL, it is possible and straightforward to rewrite the reduced basis definitions and the algorithms using the geometric normal form related to the lattice basis. Let $\mathbf{R} = \mathrm{GNF}(\mathbf{B})$ be the geometric normal form and define

- $\mathbf{R}_l = \begin{bmatrix} r_{k(l-1),k(l-1)} & \cdots & r_{k(l-1),kl} \\ \vdots & \ddots & \vdots \\ 0 & \cdots & r_{kl,kl} \end{bmatrix} \in \mathbb{R}^{k \times k}$ representing the k-blocks for $l \leqslant q$,
- $\mathbf{R}_{l,l+1} = \begin{bmatrix} \mathbf{R}_l & \mathbf{R}'_l \\ \mathbf{0} & \mathbf{R}_{l+1} \end{bmatrix} \in \mathbb{R}^{2k \times 2k}$ representing the $2k$-blocks for $l \leqslant q$.

One can write $\beta_k = \max\left(\det(\mathbf{R}_1) / \det(\mathbf{R}_2)\right)^{1/k}$ and any semi block $2k$-reduced basis satisfies the following properties.

- $\delta^k C_i \leqslant \beta_k^k C_{i+1}$ for $i = 1, \ldots, q-1$,
- $r_{ik,ik}^2 \leqslant \alpha r_{ik+1,ik+1}^2$ for $i = 1, \ldots, q-1$,
- $\mathbf{R}_1, \ldots, \mathbf{R}_q$ are HKZ reduced,

where $1/4 < \delta < 1$ and $\alpha = (\delta - 1/4)^{-1}$.

For the sake of brevity, we shall not bother with the rewrite of the definition of semi k-reduced basis or the reduction algorithms, and the further theoretical details of these algorithms shall not be dwelled upon due to the technicality of the subject.

input : Lattice Basis $\mathbf{b}_1, \ldots, \mathbf{b}_n \in \mathbb{Z}^m$ and k, δ (where $q = n/k$)

output: A semi k-reduced basis for the lattice $L(\mathrm{B})$

(initialization):

for $i = 0$ to $q-1$ do

HKZ reduce S_i

end

while $\left\|\mathbf{b}_{ik}^*\right\|^2 > \alpha \left\|\mathbf{b}_{ik+1}^*\right\|^2$ for some smallest $i < q$ do

(size reduce $\mathbf{b}_{ik+1}$ against $\mathbf{b}_{ik}, \ldots, \mathbf{b}_1$):

for $j = ik$ to 1 do

$\mathbf{b}_{ik+1} \leftarrow \mathbf{b}_{ik+1} - \left\lfloor \mu_{ik+1,j} \right\rceil \mathbf{b}_j$

(update the Gram-Schmidt coefficients $\mu_{ik+1,ik}, \ldots, \mu_{ik+1,1}$):

for $v = 1$ to j do

$\mu_{ik+1,v} = \mu_{ik+1,v} - \left\lfloor \mu_{ik+1,j} \right\rceil \mu_{j,v}$

end

end

$\mathbf{b}_{ik} \leftrightarrow \mathbf{b}_{ik+1}$

HKZ reduce S_{i-1} and S_i

end

output $\mathbf{b}_1, \ldots, \mathbf{b}_n$

Figure 5.1. Semi k-Reduction Algorithm.

input : Lattice Basis $\mathbf{b}_1, \ldots, \mathbf{b}_n \in \mathbb{Z}^m$ and k, δ (where $q = n/k$)

output: A semi block $2k$-reduced basis for the lattice $L(\mathrm{B})$

(initialization):

for $i = 0$ to $q-1$ do

HKZ reduce S_i

end

while $\left\|\mathbf{b}_{ik}^*\right\|^2 > \alpha \left\|\mathbf{b}_{ik+1}^*\right\|^2$ or $\delta^k C_i \leqslant \beta_k^k C_{i+1}$ for some smallest $i < q$ do

if $\left\|\mathbf{b}_{ik}^*\right\|^2 > \alpha \left\|\mathbf{b}_{ik+1}^*\right\|^2$ then

(size reduce $\mathbf{b}_{ik+1}$ against $\mathbf{b}_{ik}, \ldots, \mathbf{b}_1$):

for $j = ik$ to 1 do

$\mathbf{b}_{ik+1} \leftarrow \mathbf{b}_{ik+1} - \left\lceil \mu_{ik+1,j} \right\rfloor \mathbf{b}_j$

(update the Gram-Schmidt coefficients $\mu_{ik+1,ik}, \ldots, \mu_{ik+1,1}$):

for $v = 1$ to j do

$\mu_{ik+1,v} = \mu_{ik+1,v} - \left\lceil \mu_{ik+1,j} \right\rfloor \mu_{j,v}$

end

end

$\mathbf{b}_{ik} \leftrightarrow \mathbf{b}_{ik+1}$

HKZ reduce S_{i-1} and S_i

end

if $\delta^k C_i \leqslant \beta_k^k C_{i+1}$ then

HKZ reduce L_{i-1}

end

end

output $\mathbf{b}_1, \ldots, \mathbf{b}_n$

Figure 5.2. Semi Block $2k$-Reduction Algorithm.

For further information, reader is advised to refer to (Schnorr 1987, Gama, et al. 2006, Schnorr 2006a).

Remarks on the HKZ reduction:

- An algorithm to perform such HKZ reduction is firstly proposed by Kannan (1983, 1987b). Later, the algorithm is improved by Helfrich (1985), Schnorr (1987). Schnorr, in order to perform BKZ reductions, uses his improved version of Kannan's algorithm, which is a $n^{n/2+o(n)}$-time algorithm requiring a polynomial space. Furthermore, the complexity of Kannan's algorithm is further refined to $n^{n/2e+o(n)} \approx n^{0.184n+o(n)}$ by Hanrot and Stehlé (2007). As a side note, Kannan's algorithm makes use of the LLL algorithm in order to obtain good bases during computation.

- There is another algorithm to compute HKZ reduced bases due to Ajtai et al. (2001) , whose running time is provably bounded by $2^{5.9n}$ by Nguyen and Vidick (2008). However, there are two drawbacks of this algorithm. It requires an exponential space and there is a tiny probability that the computed basis is not HKZ reduced, i.e. the algorithm is probabilistic.

- HKZ reduction of the $2k$-block $\pi_{ik+1}(\mathbf{b}_{ik+1}), \ldots, \pi_{ik+1}(\mathbf{b}_{ik+2k})$ decreases $D_i = \prod_{j=0}^{i-1} C_i$ by at least a factor of δ.

Remarks on the semi reduction algorithms:

- The reduction algorithms form a hierarchy between LLL reduction and HKZ reduction. For $k = 2$, k-reduction algorithm performs LLL reduction whereas for $k = n$ the reduction becomes HKZ. In general, the algorithms are called BKZ reduction algorithms and semi block $2k$-reduction algorithm outputs relatively better bases while having the same time and space complexity.

- Notice that for $k = O(n)$, the running time gets exponential. Therefore, the algorithms perform HKZ reductions on blocks of size k to avoid an exponential run time.

Remarks on the complexity and the approximation factors for SVP:

- The main difference between the semi k-reduction and the semi block $2k$-reduction is that although they both have the same time and space complexity, the semi k-reduction produces slightly longer bases. The algorithms are feasible for $k \leqslant 25$.

- Semi block $2k$-reduction requires $O\left(n^2\left(k^{k/2+o(k)}+n^2\right)\log B\right)$ arithmetic operations on $O(n\log B)$ bit integers and achieves an approximation factor of $\left(6k^2\right)^{n/(2k)}$.

- For the choice of $k = O(\log n/\log\log n)$ the approximation factor of semi block $2k$-reduction algorithm is $2^{O\left(n(\log\log n)^2/\log n\right)}$ while the run time is still polynomial. Furthermore, if one uses the probabilistic algorithm in (Ajtai, et al. 2001) within the blocks of size k, instead of the deterministic SVP algorithm in (Kannan 1987b), the achieved approximation factor becomes $2^{O(n(\log\log n)/\log n)}$ for $k = O(\log n)$ in randomized polynomial time.

- The $\left(6k^2\right)^{n/(2k)}$ factor is initially given by Schnorr (1987). Furthermore, assuming the unproven bound $\gamma_k \leqslant k/6$ for $k \geqslant 24$, one obtains a factor of $(k/3)^{n/k}$ (Schnorr 2001, 2003).

- Ajtai (2003) proves that the best possible approximation factor for the semi block $2k$-reduction algorithm is $k^{\varepsilon(n/k)}$ for some constant $\varepsilon > 0$ under the condition $k = o(n)$. This implies that if the algorithm runs in polynomial time then we have already a polynomial time algorithm for finding the shortest nonzero vector.

- Gama et al. (2006) prove the approximation factor of $\approx (\beta_k/\delta)^{(n-1)/(2k)}$. For $k = 24$, this factor $< (1.165)^{(n-1)/2}$, moreover, as noted by (Schnorr 2006a), this can be shown to be $\approx (1.034)^{(n-1)/2}$ under the heuristic in (Schnorr 2003).

5.4. Blockwise Lattice Basis Reduction Variants

5.4.1. BKZ Variants

Schnorr and Euchner (1994) define k-reduced with δ bases by relaxing the conditions of k-reduced bases in another way. A basis $\mathbf{b}_1,\ldots,\mathbf{b}_n \in \mathbb{Z}^m$ is k-reduced with δ,

$1/2 < \delta < 1$, if it is size reduced and

$$\delta \|\mathbf{b}_i^*\|^2 \leqslant \lambda_1 \left(\pi_i \left(L\left(\mathbf{b}_i, \ldots, \mathbf{b}_{\min\{i+k-1,n\}}\right)\right)\right)^2 \tag{5.18}$$

for $i = 1, \ldots, q-1$ where $q = n/k$ and $2 < k < n$. They propose a practical BKZ variant which uses LLLFP algorithm proposed by the authors in the same paper. Furthermore, they show a BKZ reduction algorithm with pruning by presenting a rule to speed up the enumeration process in the enumeration algorithm. The authors state that the algorithm has no polynomial time proof however it behaves well in the practice.

Schnorr and Hörner (1995) develop a pruning rule based on a particular heuristic in order to speed up the enumeration of the short lattice vectors. The improvement yields another BKZ algorithm with pruning, enhancing the enumeration process of the algorithm presented in (Schnorr and Euchner 1994). The authors state that there is no polynomial runtime proof for the proposed variant and perform some applications in order to demonstrate its practicality.

5.4.2. Primal/Dual Segment Reduction

Koy (2004) introduces a basis reduction algorithm which works on k-blocks. The algorithm is named as the primal/dual segment reduction algorithm, and it runs with the parameters k, δ in $O\left(n^3 k^{k/2+o(k)} + n^4\right)$ time, needs to store $n^2 + O(n)$ lattice vectors and obtains the proven approximation factor $\left(\alpha\gamma_k^2\right)^{(n/k-1)/2}$ where in general $\alpha \approx 4/3$ and $\delta \approx 1$. The algorithm is feasible for $k \leqslant 50$, and the approximation factor becomes $(k/6)^{n/k}$ assuming the unproven bound $\gamma_k \leqslant k/6$ for $k \geqslant 24$. In general, primal/dual method achieves better approximation factors than the semi block $2k$-reduction algorithm, more accurately approximation factor $\approx 1.075^{(n-1)/2}$, for the same blocksize, since $\gamma_k^2 = \Theta(k)$. However, the semi block $2k$-reduction algorithm may become superior if is close to the lower bound $\beta_k > k/12$ given by Gama et al. (2006) making $\beta_k/\delta \leqslant \sqrt{\alpha}\gamma_{2k}$. Nevertheless, both algorithms perform equally powerful in approximation of SVP and perform even better ($\approx (1.034)^{(n-1)/2}$ approximation factor)under particular heuristic in (Schnorr 2006a). In addition, it is possible to decrease the approximation factor to $\approx (1.025)^{(n-1)/2}$ for $k = 80$ via replacing the HKZ reduction by the random sampling reduction in (Schnorr 2003) under the mentioned heuristic. The resulting algorithm is highly parallel but has no polynomial runtime proof. Also see

(Schnorr 2007).

5.4.3. $2k$-Block Rankin Reduction and Transference Reduction

Gama et al. (2006) propose a blockwise basis reduction algorithm called $2k$-block Rankin reduction which is an improvement to Schnorr's semi block $2k$-reduction algorithm. The algorithm make use of the Rankin's constant, which is the generalization of the Hermite's constant, playing a similar role to the Schnorr's constant β_k. However, this version of the algorithm is only of theoretical interest because it requires a k -Rankin reduction subroutine which solves the half volume problem exactly and the exact solution is not known for the dimensions higher than 4. In order to fill this gap, the authors propose another reduction method which is an approximation algorithm for the half volume problem in dimension $2k$, called the transference reduction. In dimension $2k$, it is cheaper than the $2k$-dimensional HKZ reduction used in the semi block $2k$-reduction algorithm. The $2k$-block Rankin reduction algorithm, when plugged with the transference reduction, which utilizes an improved reduction strategy while reducing the large blocks instead of using the HKZ reduction, performs cheaper reductions therefore allows the use of higher k. The upper bound on the approximation factor $\approx \beta_k^{n/(2k)} \lessapprox k^{n \ln 2/k}$ is reduced by $\ln 2$-th power to $\approx \gamma_k^{n/k} \lessapprox k^{n/k}$. An important point concerning this discussion is that the analysis of the Kannan's shortest vector algorithm by Hanrot and Stehlé (2007), has an effect on the approximation upper bounds of semi block $2k$-reduction algorithm of Schnorr (1987) and the reduction algorithms of Gama et al. (2006) .

5.4.4. Slide Reduction

Gama and Nguyen (2008a) present a polynomial-time blockwise reduction algorithm which finds a short vector within the Mordell's inequality , $\gamma_n \leqslant \gamma_k^{(n-1)/(k-1)}$, which is the natural generalization of the Hermite's inequality, $\gamma_n \leqslant \gamma_2^{(n-1)} = \sqrt{4/3}^{(n-1)}$ achieved by LLL (Lenstra, et al. 1982). The algorithm is called the slide reduction and achieves $(1+\varepsilon)\gamma_k^{(n-k)/(k-1)} \approx \gamma_k^{(n-k)/(k-1)}$ approximation to SVP. This factor is an improvement of those of (Schnorr 1987, Gama, et al. 2006). However, for $k = O(\log n / \log\log n)$, the approximation factor is similar, specifically $2^{O\left(n(\log\log n)^2/\log n\right)}$.

It is also possible to use the probabilistic algorithm in (Ajtai, et al. 2001) within the blocks of size k, instead of the deterministic SVP algorithm presented in (Kannan 1987b). In this case, the algorithm runs in randomized polynomial time achieving $2^{O(n(\log\log n)/\log n)}$ approximation for $k = O(\log n)$.

5.5. Brief Notes

5.5.1. Generalized Blockwise Lattice Basis Reduction

Kaib and Ritter (1994) generalizes the concept of blockwise lattice basis reduction from the l_2 norm to arbitrary norms and show that

$$\frac{4}{i+3}\kappa_k^{-2^{\frac{i-1}{k-1}}} \leqslant \|\mathbf{b}_i\| / \lambda_i \leqslant \frac{i+1}{4}\kappa_k^{2^{\frac{n-1}{k-1}}} \tag{5.19}$$

is satisfied for all norms where κ_k^2 is the generalization of the Hermite's constant γ_k.

5.5.2. Improvement on the Nearest Plane Algorithm

Schnorr (1996) improves the nearest plane algorithm, (Babai 1986), by using BKZ reduced bases.

5.5.3. Parallel Block Reduction Algorithm

Wetzel (1998) presents a new parallel block-reduction algorithm and thus a hierarchy of parallel lattice basis reduction algorithms between the known parallel all-swap algorithm which is a parallelization for block-size two (or the LLL algorithm) and the reduction algorithm for block size equals the lattice dimension which corresponds to the known sequential lattice basis reduction algorithm.

5.6. Referential

For more information on HKZ bases refer to (Lagarias, et al. 1990, Schnorr 1996, Blömer 2000, Pendavingh and van Zwam 2007, Hanrot and Stehlé 2008). Also detailed explanations of blockwise lattice basis reduction algorithms and comparisons between

the most practical reduction algorithms can be found in (Gama, et al. 2006, Schnorr 2006a, 2007, Gama and Nguyen 2008a,b).

CHAPTER 6

LINKING CRYPTANALYSIS

This chapter aims to provide more insight on the concept of lattice basis reduction in cryptanalysis. To accomplish this, the meaning of good basis and how a good basis is used to (approximately) solve the shortest and the closest vector problems are outlined. Furthermore, a natural and general cryptographic scheme appearing in the lattice based cryptosystems is introduced to demonstrate the straightforward use of lattices in the construction of cryptographic schemes. In addition, two most common heuristics which are used in the cryptanalysis of the lattice based cryptosystems are explained. Following this discussion, several lattice based attacks emphasizing the important role of the lattice basis reduction as a cryptanalytic tool are described and the chapter is concluded.

6.1. The Role of the Good Bases

6.1.1. Good Bases

As stated before, there is no unique definition of good basis. In general in order to obtain a good or a small basis one could search for a basis where the maximum of the lengths of its vectors is minimized or one could look for a basis where the product of the vector lengths is minimized. For the sake of concreteness, one can define two constants related to a given lattice. Let $\mathbf{B}$ be a lattice basis and let Λ be the lattice defined by $L(\mathbf{B})$ of dimension n and degree m. Then, if $\mu(\mathbf{B}) = \max_{i=1}^{n} \|\mathbf{b}_i\|$ is regarded as a measure for the length of a basis, one can define the smallest basis length in the lattice as

$$\mu(\Lambda) = \min \mu(\mathbf{B}) \tag{6.1}$$

where $\mathbf{B}$ runs over all bases for the lattice Λ. Furthermore, if $\delta(\mathbf{B}) = (\prod_{i=1}^{n} \mathbf{b}_i / \det(\Lambda))^{1/n}$ is the normalized orthogonality defect of a basis, it is possible to define the minimal orthogonality defect of the basis in the lattice as

$$\delta(\Lambda) = \min \delta(\mathbf{B}) \tag{6.2}$$

here $\mathbf{B}$ runs over all bases for the lattice Λ. The search for a good basis requires to find a basis $\mathbf{B}$ where either $\mu(\mathbf{B})$ is as close to $\mu(\Lambda)$ as possible or $\delta(\mathbf{B})$ is as close to $\delta(\Lambda)$ as it can get. Furthermore, it is possible to extend the definition of the normalized orthogonality defect and the minimal orthogonality defect to the set of n linearly independent lattice vectors instead of just the basis vectors.

In general, lattice basis reduction algorithms, especially LLL, tend to find more orthogonal and shorter lattice basis than the given input basis, and as suggested by the experimental results they perform better than the proven worst cases.

For a more detailed discussion on the lattice problems related to the basis reduction, see (Micciancio and Goldwasser 2002).

6.1.2. Good Bases and Lattice Problems

In general, to solve the shortest or the closest vector problem, one needs to solve the more general problem of lattice basis reduction. The importance of the quality of the lattice basis in the solution of the hard lattice problems such as the shortest and the closest vector problems can be demonstrated simply as follows.

Let the basis $\mathrm{B} = \{\mathbf{b}_1, \mathbf{b}_2\}$ and $\mathrm{C} = \{\mathbf{c}_1, \mathbf{c}_2\}$ be the two different bases of the same lattice as shown in the Figure 6.1. Using the figure, one can immediately note that

- $\mu(\mathrm{B}) < \mu(\mathrm{C})$,
- $\delta(\mathrm{B}) < \delta(\mathrm{C})$.

Considering the shortest vector problem, the basis B yields a better approximation to the shortest vector problem. In fact, it can be inferred from the figure that B contains a shortest vector of the lattice. Whereas, the basis C approximates the shortest vector to within a factor of around ≈ 5.

In order to find the approximate or the exact closest vector to a given target vector, the fundamental parallelepipeds related to the lattice bases are used. Let B be the any basis of a given lattice, then the fundamental parallelepiped, or the fundamental domain, related to the basis B is defined as

$$F(\mathrm{B}) = \left\{ \mathbf{Bx} : \mathbf{B} \in \mathbb{R}^{m \times n}, x_i \in [0,1] \right\}. \tag{6.3}$$

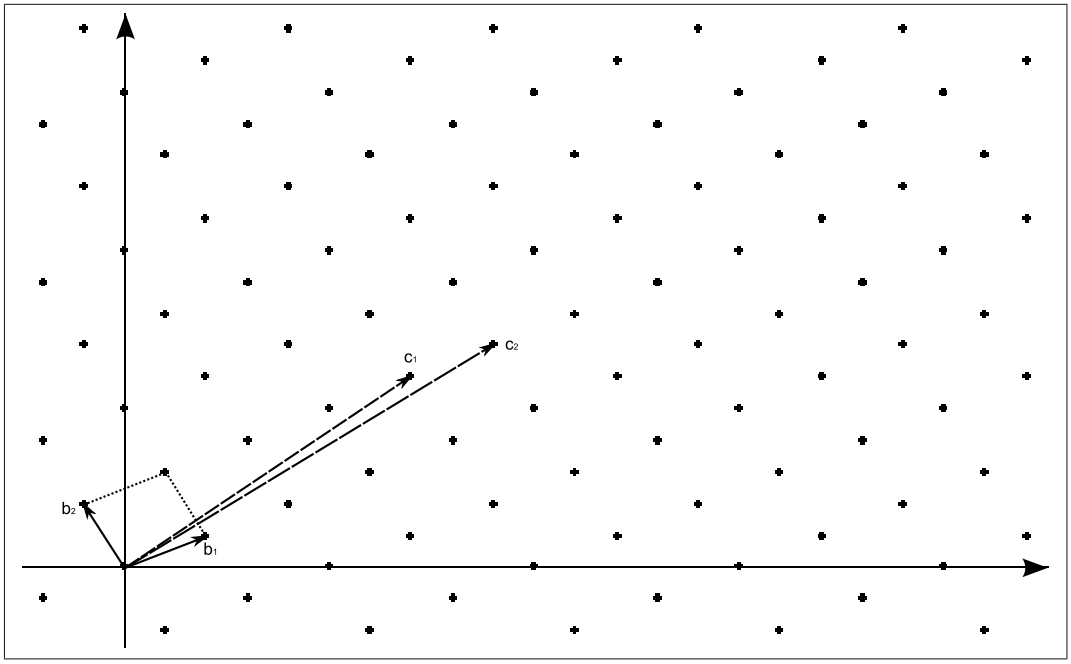


Figure 6.1. Good and Bad Bases.

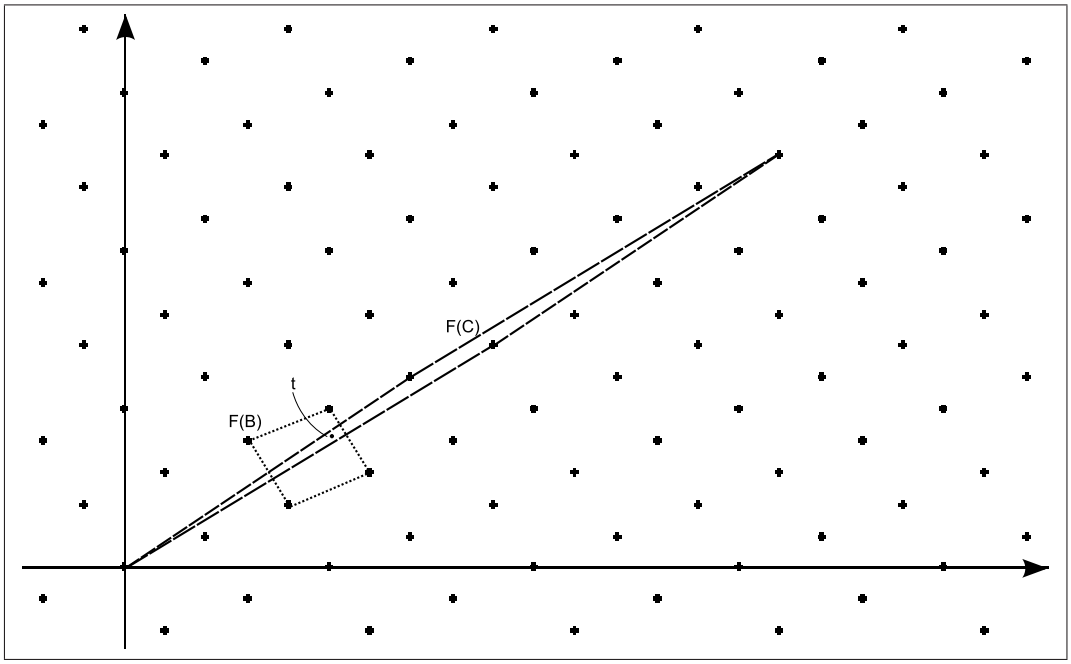


Figure 6.2. Translated Fundamental Parallelepipeds.

Now consider Figure 6.2. Given the target vector **t**, in order to find the approximate closest lattice vector to the target vector, the fundamental parallelepipeds related to the bases B and C are translated to wrap the target vector **t**. The closest vector is approximated as the vertex of the translated fundamental parallelepiped which is closest to the target vector. It is obvious that, for the given target vector in the figure, the close vector approximated using the lattice basis C is farther than the one approximated using the basis B. In fact, the latter vector is the closest lattice vector to the given target vector.

6.2. A General Trapdoor Outline

Given a basis for a lattice, it is easy to generate a vector which is close to the lattice. This can be done by generating a lattice vector using the basis and adding some carefully chosen vector to obtain a close vector. However, as implied by the closest vector problem and it variants, to recover the closest lattice vector is not an easy task using an arbitrary lattice basis. Even the approximations, i.e. the close vectors, are not sufficient in case the given arbitrary basis is not reduced properly or a good enough basis can not be found.

For this trapdoor, there are two possibilities to encode the message which can be stated abstractly as follows. Let **v** be the generated lattice vector and **d** be the added vector which does not belong to the lattice, then a close vector can be obtained as $\mathbf{t} = \mathbf{v} + \mathbf{d}$. At this point, one can encode the message inside the lattice vector **v** or inside the chosen non lattice vector **d** which is sometimes called the error vector. In both approaches, additional steps are performed to properly, and securely, encrypt the plaintext and decrypt the ciphertext.

The above scheme is outlined here because it poses a clear and an obvious trapdoor which is easy to understand, reflecting the framework of some of the popular lattice based cryptosystems such as GGH (Goldreich, et al. 1997) and NTRU (Hoffstein, et al. 1998, 1999). However, the lattice problem on which the GGH cryptosystem is based is the closest vector problem whereas the NTRU is based on the shortest vector problem and the closest vector problem. Furthermore, it is important to note that, in general, this kind of raw trapdoors are not sufficient to built full fledged cryptosystems without taking further and proper measures to minimize, or if possible remove, the inherent flaws and

the decryption errors (if the scheme is probabilistic) etc., while maintaining the performance / security trade off at a practical level. These measures are generally composed of the specific parameters sets and values together with additional and auxiliary functionality, which almost always change from one construction to another affecting the overall system.

It is not necessary to use a similar approach to the one described above in order to construct a lattice based cryptosystem. For example, Ajtai and Dwork (1997) propose such a cryptosystem (the AD cryptosystem) which is based on a variant of the shortest vector problem where the lattice gap λ_2/λ_1 is polynomial in the dimension. Although based on a lattice problem, the use of lattices in the AD cryptosystem is not as explicit as the above scheme. The significance of the AD cryptosystem is that it enjoys an interesting security proof on the worst case hardness of the underlying shortest vector problem and this type proofs are not present in other lattice based or non lattice based public key cryptosystems in general.

As a remark, it should be mentioned that the AD and GGH cryptosystems are broken for all choice of parameter values where the schemes are practical, whereas the NTRU cryptosystem still remains unbroken after being subject to slight changes over time.

6.3. Important Heuristics in Practice

6.3.1. Gaussian Heuristic

In a full rank lattice with dimension n, the number of lattice points in the centered ball of radius r at the origin can be approximated as the number of fundamental parallelepipeds inside the ball. Thus, the number of such lattice points corresponds to

$$\approx \frac{\mathrm{vol}\left(B_n\left(\mathbf{0},r\right)\right)}{\mathrm{vol}\left(P\left(\mathrm{B}\right)\right)} \tag{6.4}$$

This approach is more justifiable if the ball is centered at the origin, the radius grow to infinity and the dimension (or degree) of the lattice is sufficiently large. If one chooses the radius so that

$$\mathrm{vol}\left(P\left(\mathrm{B}\right)\right) = \det\left(L\left(\mathrm{B}\right)\right) \approx \mathrm{vol}\left(B_n\left(\mathbf{0},r\right)\right) \tag{6.5}$$

where, heuristically, it can be deduced there is a non zero lattice point in the ball. Therefore, the expected length of the shortest vector in the lattice is less than the radius of the ball. The volume of the n-dimensional ball or radius r is

$$\mathrm{vol}\left(B_n\left(\mathbf{0},r\right)\right) \approx \left(\frac{2\pi e}{n}\right)^{n/2} r^n \tag{6.6}$$

and when the volume of the ball and the lattice are equal,

$$\det\left(L\left(\mathrm{B}\right)\right) \approx \left(\frac{2\pi e}{n}\right)^{n/2} r^n \tag{6.7}$$

Solving the approximate equality for the radius r yields

$$r \approx \sqrt{\frac{n}{2\pi e}} \det\left(L\left(\mathrm{B}\right)\right)^{1/n} \tag{6.8}$$

Therefore, according to Ajtai (2002), a random full rank lattice Λ satisfies the following properties based on the Gaussian heuristic with an extremely high probability.

- $\lambda_1 = \lambda_1\left(\Lambda\right) = \min\limits_{\mathbf{v}\in\Lambda-\{\mathbf{0}\}} \|\mathbf{v}\| \approx \sqrt{\frac{n}{2\pi e}} \det\left(\Lambda\right)^{1/n}$
- $\min\limits_{\mathbf{v}\in\Lambda-\{\mathbf{0}\}} \|\mathbf{v}-\mathbf{t}\| \approx \sqrt{\frac{n}{2\pi e}} \det\left(\Lambda\right)^{1/n}$ where $\mathbf{t}$ is a random target vector.

Furthermore, with a high probability, the length of the shortest vector in a lattice lies between

$$\sqrt{\frac{n}{2\pi e}} \det\left(\Lambda\right)^{1/n} \quad \text{and} \quad \sqrt{\frac{n}{\pi e}} \det\left(\Lambda\right)^{1/n} \tag{6.9}$$

See (Hoffstein, et al. 1998, Silverman 2006).

6.3.2. Kannan's Embedding Heuristic

An important strategy in practice to solve the closest vector problem is the Kannan's embedding method (Kannan 1987b, Nguyen 1999, Micciancio and Goldwasser 2002). The method is not provable in general and therefore may fail to find the closest vector. It can be used when the target vector is very close to the lattice to find a closest vector by using an algorithm to solve the (unique) shortest vector problem. The outline of the method is as follows.

Given a n dimensional and m degree lattice with the basis $\mathrm{B} = \{\mathbf{b}_1,\ldots,\mathbf{b}_n\}$, and a target vector $\mathbf{t} \in \mathbb{R}^m$, it is possible to embed the n dimensional lattice in $\mathbb{R}^m$ to another lattice with $n+1$ dimension in $\mathbb{R}^{m+1}$. The new lattice is spanned by the basis

$\{(\mathbf{b}_1,0),\ldots,(\mathbf{b}_n,0),(\mathbf{t},c)\}$ where the coefficient c is determined depending on the lattice (in many cases $c=1$). Experiments suggest that

- the shortest vector of the new lattice is in the form $(\mathbf{t}-\mathbf{v},c)$ where $\mathbf{v}$ is a closest vector to $\mathbf{t}$ in the original lattice,
- the second minimum of the new lattice corresponds to the first minimum of the original lattice,

if the distance to the lattice is smaller than the length of the shortest vector. In fact, to recover $(\mathbf{t}-\mathbf{v},c)$ it is sufficient to approximate the shortest vector problem within a factor of $\lambda_1/\|\mathbf{t}-\mathbf{v}\|$. Also see (Fischlin and Seifert 1999).

6.4. Cryptanalytic Considerations and Lattice Attacks

6.4.1. Brief Considerations

In the lattice based public key cryptosystems, the typical elements which give rise to the cryptanalytic weaknesses can roughly be listed as follows.

- The structure of the lattice used.
- The choice of systems parameters.
- The key generation, encryption and decryption processes.
- The non-hardness of the underlying problem.

Although listed separately, these issues are closely related to each other. Generally speaking, there are some certain classes of lattices which seems to be more secure compared to the other classes. The failure in the selection of the appropriate lattice results in the exploitation of the lattice structure in terms of cryptanalytic attacks. The properties of the selected lattice are also related to the proper choice of the system parameters which in almost all cases directly affects the level of security. Furthermore, perhaps more than the other contemporary public key cryptosystems, the lattice based public key cryptosystems, relies on the properly determined parameters. These parameters have great impact on the practicality and the security of the system. In addition, being related to the

chosen runtime parameters, the flaws in the key generation, encryption and decryption processes may compromise the security of the cryptosystem seriously. It might be possible to reduce the hard problem instance used in the process to an easier instance of the same or another problem, after performing some straightforward computations due to the flawed mechanism. Another important thing is whether the hardness assumption of the problem which is used in the construction of the trapdoor is valid or not. For example, although a certain problem has hardness proof, it does not mean that all instances of the problem are equally hard or all the variants of the problem satisfy the same hardness level or conditions. Therefore, assuming that the underlying problem is hard in case it is not comes at the expense of practicality, performance and security.

6.4.2. Lattice Attacks

Aside from the general cryptanalytic attacks, there are also lattice attacks on the lattice based public key cryptosystems, striving to solve approximately or exactly the underlying lattice problem which is in general the shortest vector problem or the closest vector problem or a variant of any of them. Although, being different from one lattice based public key cryptosystem to another, it is possible to describe the lattice attacks simply as follows.

6.4.2.1. Lattice Attacks on the Private Key

The private key described in the aforementioned scheme is a good basis of the lattice. It might be possible to recover a good basis by using the reduction algorithms. However, in general, such approaches are not practical due to the system specifications.

On the other hand, sometimes it is possible to represent the private key as a short vector in the lattice for practicality. The key still functions as a good lattice basis however it is just more compact. A similar case arises in NTRU (Hoffstein, et al. 1999). The underlying problem is the approximate shortest vector problem where the private key is encoded in a short lattice vector in a lattice which is slightly modified form of the original. The lattice basis reduction algorithms can be used to recover the private key after performing some extra computations to minimize the recovery time such as

balancing the lattice.

It is important to note that, to obtain some insight on the vulnerability of the private key, and thus on how the basis reduction algorithms will perform, measuring of how far a lattice differs from a random lattice is a good approach. In case the private key is a short vector, the Gaussian heuristic is useful to do this.

6.4.2.2. Lattice Attacks on the Message

It is expected that the vulnerability of message should be close to the vulnerability of the private key so that the basis reduction or the lattice attacks will have equally difficult time and there will be nontrivial exploits.

It might be possible to recover the message by solving the closest vector problem with some proper measures. In general, the cryptosystems are specified in a such way that exactly solving the underlying problem (at least directly) is highly impractical and requires extensive computation. At this point, the reduction algorithms come handy. It might be possible to combine the inherent flaws in the encryption process with the Kannan's embedding heuristic outlined before and solve the shortest vector approximately and retrieve the close lattice vector where the message is encoded. This is the way how Nguyen (1999) breaks the GGH cryptosystems in its original form in (Goldreich, et al. 1997), using two weaknesses related to the encryption process where the first one is the relatively short length of the error vectors compared to the lattice vectors and the other is the flawed way of choosing the entries of the error vectors. The first weakness causes the basis reduction algorithms to be applied practically to recover the close vector and the second one allows the simplification in the underlying closest vector problem. Although, according to the practical behaviour of the reduction algorithm used, it might be possible to break the cryptosystem even if no serious flaws are existent.

In the NTRU cryptosystem, a similar situation to the one described in recovering the NTRU private key arises. The message vector can be encoded in a short lattice vector in a slightly different lattice and lattice basis reduction algorithms can be applied to solve the approximate shortest vector problem.

6.4.2.3. Lattice Attacks on a Spurious Key

This approach is firstly proposed for the cryptanalysis of the NTRU cryptosystem in (Coppersmith and Shamir 1997). It requires finding a sufficiently short vector which will as the private key during the decryption rather than finding the private key itself. In particular, it can be possible to find several vectors (not as short as the private key) which can be used to decrypt the message partially. Then, these partial decryptions are pieced together to get the full decryption. This approach is practical if the time required to find the spurious key is significantly less than the time required to find the private key.

6.4.2.4. An Example of Exploiting the Lattice Structure

May (1999) exploits the structure of the special class of lattices used in the NTRU cryptosystem and proposes a lattice based attack. The attack aims to make the lattice gap as big as possible so that heuristically the approximation to the shortest vector within a factor will yield the shortest vector itself, i.e. the private key. To do this, another class of lattices, which have reduced dimension and satisfy some certain conditions, are inferred from the Coppersmith-Shamir lattices, see (Coppersmith and Shamir 1997, Hoffstein, et al. 1999). Furthermore, the generalization of this approach is also discussed in (May and Silverman 2001).

Note that the discussed attacks are in general tightly related to the structure of the particular cryptosystems. However, in all attacks, the lattice basis reduction algorithms play crucial role together with some certain heuristics. Furthermore, it might also be possible to improve the general cryptanalytic attacks, such as the chosen ciphertext attacks, the partial information attacks and the brute force attacks, using the lattice algorithms depending on the chosen lattice based public key cryptosystem.

6.5. Referential

For more information on lattice based cryptography and improvements refer to (Ajtai and Dwork 1997, Dwork 1998, Goldreich, et al. 1997, Hoffstein, et al. 1999, Cai and Cusick 1999, Sakurai 2000, Micciancio 1999, 2001a, Regev 2004b, 2005, Ajtai

2005, Paeng, et al. 2002, Kawachi, et al. 2007, Peikert and Waters 2007, Gentry, et al. 2007, Micciancio 2007). The cryptanalysis of and lattice based attacks on the various lattice based cryptographic constructions can be found in (Coppersmith and Shamir 1997, Nguyen and Stern 1998, May 1999, Nguyen 1999, Gentry 2001, Gentry and Szydlo 2002, Nguyen and Regev 2006, Han, et al. 2007). Furthermore, lattices are important tools for the cryptanalysis of non lattice based constructions and details can be found in (Joux and Stern 1998, Nguyen 2001, Micciancio 2002, Nguyen 2008). As a last note, reader should refer to (Gama and Nguyen 2008b) for significant assessments on how lattice basis reduction methods behave in practice arousing security and efficiency implications on the cryptanalytic and cryptographic perspectives.

CHAPTER 7

AN EVALUATION FOR EXPERIMENTS

As mentioned before, the lattice basis reduction algorithms, in practice, perform much better than the expected (and proved) worst case bounds in terms of running time and output quality. This phenomenon affects the understanding of the current concepts of the basis reduction and the reduction algorithms, and shows that more insight should gained on what is really going on behind these algorithms. The aim of this section is to provide practical assessments on the most popular variants of the LLL and BKZ lattice basis reduction algorithms. To do this, the works of Nguyen and Stehlé (2006), Gama and Nguyen (2008b) are exploited.

7.1. Importance of the Practical Assessments

Assessments of the practical performance of the lattice basis reduction algorithms are necessary and important for many reasons. Some of these reasons can be stated as follows.

- Although experimental, it has been a fact that the lattice basis reduction algorithms perform much better in practice.
- The use of various heuristics in different algorithmic variants brings forth the necessity of the experimental measurements.
- New heuristics, which can be used to further speed up the reduction process, may be revealed, playing a role in the proposition of new reduction schemes.
- The experimental measurements and observations may reveal new information on the practical hardness of the main lattice problems and enable the comparison of the various computational hardness assumptions used in theoretical lattice-based cryptography.
- The observations may also reveal some classification among different lattice

classes, such as hard and easy lattice instances. Thus, several adjustments can be performed on the choice of parameters in the cryptographic schemes in order to strengthen the security level of the instance.

- The practical assessments are likely to allow discovery of the differences and similarities between the practical and the theoretical behaviour of the reduction algorithms.

- The assessments can be used to draw the limits on what is achievable by the use of today's algorithms.

Furthermore, it is important to note that the most commonly used reduction algorithms in practice are not the classical version of the LLL algorithm (Lenstra, et al. 1982) and the BKZ algorithm (Schnorr 1987). Instead, the use of variants of these algorithms proposed in (Schnorr and Euchner 1994) are still preferred widely by cryptanalysts and mathemticians etc, because these algorithms have managed to uphold their practical superiority against the many other heuristic and theoretic variants which have been introduced since then.

7.2. Preliminary Remarks

Nguyen and Stehlé (2006) are the firsts to give a detailed and clearer explanation on the average case behaviour of the LLL reduction algorithm by performing extensive experiments and proposing heuristic arguments assessing the practicality of the algorithm (and some of its variants). Furthermore, Gama and Nguyen (2008b) extend the study to the most popular reduction algorithms in the practice, namely LLL (Lenstra, et al. 1982), LLL with deep insertions (Schnorr and Euchner 1994) and BKZ variant in (Schnorr and Euchner 1994). The authors discover very important experimental results on the limits of what today's most practical reduction algorithms can achieve. Shortly, they show that the lattice basis reduction algorithms are very optimistic in practice and state that the current algorithms, achieve the approximation factor of $\leqslant 1.01^n$ on the average and $\leqslant 1.02^n$ in the worst case. Furthermore, it is concluded that to prevent reduction algorithms to take advantage of the lattice structure of the worst cases, dimension should be very high.

In order to perform solid experiments, one needs to use mathematically sound concept of randomness in lattices. Such concepts are also discussed and utilized in (Nguyen and Stehlé 2006, Gama and Nguyen 2008b), and will shortly be introduced here.

The concept of random lattices is a mathematically sophisticated subject. As stated by Nguyen and Stehlé (2006), one might select a randomly generated lattice which is used in a particular application such as algorithmic number theory or cryptography. However, it should be noted that this type of lattices may not be random in the mathematical sense. For example, in cryptanalysis, lattices to which reduction algorithms are applied generally have their first minimum much shorter than all the other minima. Nevertheless, it can be beneficial to restrict the selection of lattices from a particular class to discuss the practicality of the reduction algorithms in the problems where the chosen particular subset is significant. Ajtai (2002), Goldstein and Mayer (2003) provide efficient means to generate lattices in a mathematically random sense. The random lattices satisfy the following approximate equality based on the Gaussian heuristic with an extremely high probability,

$$\frac{\lambda_1(L)}{vol(L)^{1/n}} \approx \frac{\Gamma(1+n/2)^{1/n}}{\sqrt{\pi}} \approx \sqrt{\frac{n}{2\pi e}}. \tag{7.1}$$

Once a lattice is selected, one needs to choose a lattice basis out of infinitely many possible choices in random. In this case, however, there is no clear definition of a random basis. It may be expected that the random basis consists of long vectors and is not reduced. As a random basis, one might use the Hermite normal form (HNF) of the lattice since it gives relatively less information on the lattice and can be computed in polynomial time. However, HNF of some lattices may have some special properties (for example, the HNF of NTRU lattices (Hoffstein, et al. 1998) are reduced in some sense). Another option could be to use the heuristic approach of Goldreich et al. (1997) which is the transformation of the secret basis into an equivalent large public basis by randomly multiplying with the generators of special linear group over the set of integers, i.e. the set of integer matrices with determinant 1. Unfortunately, in this approach, it is difficult to control the size of the entries and theoretical results are hard to obtain. On the other hand, a promising and less heuristic approach is presented in (Nguyen and Stehlé 2006). The approach considers the full-dimensional integer lattices. If the norm bound of the basis is much greater than the n-th root of the lattice determinant, i.e. $B >> (\det\Lambda)^{1/n}$,

it is possible to sample efficiently and uniformly lattice points whose norms are bounded by B. (Ajtai 1996) (Klein (2000) also proposes a sound way to sample lattice vectors, see also (Gentry, et al. 2007, Nguyen and Vidick 2008)). The sampled points are linearly independent with an extremely high probability, however they do not necessarily form a basis for the lattice. To lift this set of linearly independent vector to a lattice basis, (Ajtai 1996) or lemma 7.1 of (Micciancio and Goldwasser 2002) can be used. The produced basis has the norm bound $(\sqrt{n}/2)\,B$.

Another important notion is the random reduced basis. Nguyen and Stehlé (2006) discuss different concepts of random LLL reduced bases. It seems convenient to adapt their practical definition. The procedure is to choose a random lattice and select a random basis, and then apply the decided reduction algorithm to obtain the reduced basis. In mathematical sense, the reduced basis may not be random. The selection of a random basis is crucial, since it affects the outcome and the resulting distribution. If an already reduced basis is selected as a random basis, the output will be atypical.

7.3. Experimental Details

The main purpose of the remaining sections is to perform experiments on the most popular variants of the LLL and BKZ lattice basis reduction algorithms in order to observe the effects of the common parameter on the their practical behaviour in terms of running time and the achieved SVP approximation factor constant. To do this:

- Random knapsack lattices are used. The basis vectors are selected as the rows of the randomly generated $n \times (n+1)$ matrices of the form

$$\begin{bmatrix} \mathbf{b}_1 \\ \mathbf{b}_2 \\ \vdots \\ \mathbf{b}_n \end{bmatrix} = \begin{bmatrix} k_1 & 1 & 0 & \dots & 0 \\ k_2 & 0 & 1 & \dots & 0 \\ \vdots & 0 & 0 & \ddots & 0 \\ k_n & 0 & 0 & \dots & 1 \end{bmatrix} \tag{7.2}$$

  where $\|\mathbf{b}_i\| \leqslant B$ for $i = 1, \dots, n$.

- The size parameter $\log B$ is kept fixed, $\log B \approx 128$ bits.

- Gaussian heuristic is used to calculate the SVP approximation factor constant,

which is defined as

$$a = \left(\frac{\|\mathbf{b}_1\|}{\lambda_1(L)} \right)^{1/n} \approx \left(\frac{\|\mathbf{b}_1\|}{vol(L)^{1/n}} \sqrt{\frac{2\pi e}{n}} \right)^{1/n} \tag{7.3}$$

Furthermore, the following libraries are benefited from in order to perform the mentioned tasks.

- NTL [1] version 5.4.2 is used to perform experiments on variants of the most practical the lattice basis reduction algorithms, namely LLL_XD, LLL_XD with deep insertions (LLL_XD_DEEP) and BKZ_XD which uses Gram-Schmidt orthogonalization and extended exponent doubles (see the library documentation).

- FPLLL [2] version 2.1.6 is used for random lattice basis generation. However, the basis generation code is slightly modified and extended for the process.

- GMP [3] version 4.2.2 and MPFR[4] version 2.3.1 are used for the underlying big integer and floating point number arithmetic.

As for the experimentation environment, a personal computer with P4 (Dual Core) 1.8Ghz 64 bit CPU, 1 GB RAM and Fedora Core 7 operating system is used. The obtained results are collectively given in Chapter 8 as conclusions.

[1] NTL, NTL: A Library for doing Number Theory, http://www.shoup.net/ntl/ (accessed May 5, 2008)

[2] FPLLL, The FPLLL Library, http://perso.ens-lyon.fr/damien.stehle/english.html#software (accessed May 18, 2008)

[3] GMP, GNU MP: The GNU Multiple Precision Arithmetic Library, http://gmplib.org/ (accessed May 20, 2008)

[4] MPFR, The Multiple Precision Floating-Point Reliable Library, http://www.mpfr.org/ (accessed May 20, 2008)

CHAPTER 8

CONCLUSION

In this chapter, the experimental results are demonstrated considering the remarks made in the previous chapter. Furthermore, some comments are provided regarding the complementary studies which can be performed in the future.

8.1. Observations and Results

This section presents the results of the experiments which are performed by selecting the *delta* parameter among the set of values $\{0.5, 0.51, 0.6, 0.7, 0.8, 0.9, 0.99\}$. Furthermore, during the experiments

- The *deep* parameter of the LLL with deep insertion algorithm is ranged from 10 to 50 in increments of 10.
- The *blocksize* parameter of the BKZ algorithm is varied from 10 to 30 in increments of 5.
- The dimension parameter n is ranged from 50 to 300 in increments of 50 in case of LLL and LLL with deep insertion algorithms, and from 50 to 100 in increments of 10 in case of BKZ algorithm.

8.1.1. LLL_XD

The running times and approximation factor constants achieved by LLL_XD are given in Figures A.1 and A.2 respectively.

The running time of LLL_XD can be bounded by a polynomial function in n. This can be derived from Figure A.1(a). The $\log time$ values, i.e. the timings plotted in logarithmic scale, in the figure have decreasing slopes, which implies a polynomial runtime in n. Furthermore, Figure A.1(b) shows that, although the running time seems to be exponential in *delta* when the logarithmic plot is examined, in practice, the effect

of *delta* on the running time of LLL_XD is relatively less significant especially when n gets larger since the speed up values for decreasing *delta* are greater in lower n values.

The SVP approximation factor constants achieved by LLL_XD are given in Figure A.2. For fixed *delta*, a values seem to converge to as n increases. In addition, as shown in Figure A.2(b), a values decrease as with increasing *delta* for fixed n. The difference between achieved a values for high and low n (*delta*) is remarkable and these values get closer as *delta* (n) increases. One may find the decline in the approximation factor constants unexpected, however it has a straightforward explanation. In case of fixed n, the value of a is directly controlled by $\|\mathbf{b}_1\|$, and $\|\mathbf{b}_1\|$ decreases as *delta* increases. On the other hand, when *delta* is fixed, the value of a is mainly controlled by n. Since size parameter $\log B$ is fixed, $\|\mathbf{b}_1\|$ has no apparent monotonic behaviour, and, in general, the increasing effect of the $vol\,(L)^{1/n}$ is slightly dominated by the decreasing effect of $\sqrt{n}$ Even if the value of $a^n = \left(\|\mathbf{b}_1\| / vol\,(L)^{1/n} \sqrt{2\pi e/n}\right)$ tends to increase, this increase is not sufficient to overcome the effect of taking the n-th root.

8.1.2. LLL_XD_DEEP

Figures B.1 through B.6 summarize the main parameters and their effect on the runtime of LLL_XD with deep insertions (or shortly LLL_XD_DEEP).

Figures B.1 and B.2 show that the running time of LLL_XD_DEEP is polynomial in n since the slopes of log graphs in the figure have decreasing tendencies. One might note that, in higher n, the curves in the figure tend to be linear, which is also more apparent when *delta* is higher. Nevertheless, this is not sufficient to state that the running time is be exponential or super-polynomial. Therefore, further experiments can be performed by increasing n in smaller increments in order to draw stronger conclusions.

The previous literature, such as (Gama and Nguyen 2008b), on the subject states that LLL_XD_DEEP has running time exponential in *deep*. However, according to the observations shown in Figures B.3 and B.4, the running time can be bounded by a polynomial function in *deep* for random knapsack lattices. In addition, it can be noted that when n gets larger, the shape of the timing curves becomes more similar to those of the exponential curves. This is more apparent for high *delta*. These arguments arouse the question whether the knapsack lattices are easier to reduce on the average.

The effect of $delta$ on the runtime can better be observed in the graphs in Figures B.5 and B.6. It is shown that, for fixed fixed $deep$ and n, the runtime of LLL_XD_DEEP seem to be $O(-1/\log(delta))$. The graphs show that a take-off seems to occur after around $delta = 0.90$, and it is more apparent and sharper for larger $deep$ and/or n values. Further experiments yield the fact that this take-off generally occurs around $delta \approx 0.96$.

The SVP approximation factor constants, a values, achieved by LLL_XD_DEEP with respect to the parameters n, $deep$ and $delta$ are give in Figures B.7 through B.12.

Figures B.7 and B.8 show that the achieved approximation factor constants decrease and slowly converge to 1 as n increases. However, it should be noted that there is a frequently observed exceptional circumstance when n is sufficiently low. In the experiments, it is observed that, LLL_XD_DEEP for $n = 50$ achieves better a, than those achieved by higher n for almost all fixed $deep$ and $delta$ values . The achieved constants may even beat those of $n = 300$. This can be justified as follows. As n increases, a^n increases as usual. However, since a^n values are already very low for sufficiently small n such as $n = 50$, the effect of taking $1/n$-th power does not suffice to obtain lower a values for bigger n than those of sufficiently small n. Thus, for such small n, lower a values are calculated. This occurrence is more visible when $deep$ becomes larger. However, as $deep$ increases, one needs higher $delta$ to beat larger n. On the other hand, for $n \geqslant 100$, the steady decrease in a values can be explained as in the discussion for LLL_XD in the previous section.

The SVP approximation factor constants are also noted to be convergent with $deep$ as implied by Figures B.9 and B.10. Unlike they do with respect to n, a values converge relatively faster with respect to $deep$. It can bee seen from the figure that, for fixed n and $delta$, increasing $deep$ value after some certain point may not help to improve the achieved approximation factor constants practically. If existent, this certain point varies according to the chosen n and $delta$, and it seems to inrease for higher values of n and $delta$.

As for the effect of $delta$ on the achieved a values, Figures B.11 and B.12 can provide some insight. Obviously, if one increases $delta$, smaller a values are obtained. Furthermore, the decreasing a curves seem to be linear except for $n = 50$, which is already discussed above in this section.

Considering the discussion made so far, the following remarks hints on the pos-

sible means to exploit the results of the experiments.

- Instead of using high end $delta$ values, one may choose to relax $delta$ and increase the value of $deep$ parameter to possibly achieve approximation factor constants which are as good or slightly worse. However, in this process, it is important to keep in mind that raising $delta$ beyond ≈ 0.96 greatly increases the running time, and increasing $deep$ beyond some certain point may not improve the achieved a values, while causing a possible decline in the runtime performance.

- Another way to achieve better results is to increase the dimension of the lattice. If one can devise a way to efficiently embed a given low dimensional lattice to that of high dimension, it might be possible to obtain better approximation factor constants. The crucial point is that, during the embedding process, particular lattice properties, such as the value of the size parameter and the type of the lattice etc., should remain unchanged, although it might be possible to tolerate slight changes.

- It is important to note that it might be possible to combine the remarks made above. While performing these kind of attempts to increase the runtime performance or the achieved approximation factor constants, one should carefully choose the parameter values after measuring the trade-offs and thoroughly analyzing the problem at hand so that unexpected outcomes can be avoided.

8.1.3. BKZ_XD

The running time of BKZ_XD with respect to n, $blocksize$ and $delta$ parameters are shown in Figures C.1 through C.6.

It is shown in Figures C.1 and C.2 that the running time of BKZ_XD is polynomial in n when $blocksize$ parameter is low, such as $blocksize < 20$. However, as it is further increased, running time for low and high end $delta$ values begin to get super-polynomial. It can be seen that for $blocksize > 20$, BKZ_XD is exponential in n for $delta = 0.5$, and for $blocksize = 30$, BKZ_XD is exponential for $delta = 0.5, 0.6, 0.99$. One should also note that no measurements are taken for $delta = 0.5, 0.6$ when $blocksize = 30$. This implies that BKZ_XD is highly impractical for small when $blocksize$ is high.

Running time of BKZ_XD seem to be super-exponential in *blocksize* for all values used in the experiments. When Figures C.3 and C.4 are examined, it can be observed that there is a sudden increase in the running time after $blocksize \approx 20$. For low end *delta* values, such as 0.5 and 0.6, the increase is more apparent and occurs at relatively smaller *blocksize* values. In addition, for larger n, the increase for high end *delta* values, such as 0.99, also shows significant growth. One can also derive that BKZ_XD for $blocksize \geqslant 30$ is highly impractical for low and high end *delta* values in high dimensions.

Figures C.5 and C.6 show the effect of *delta* in the running time of BKZ_XD. It is clear that as *delta* gets closer to 1, the running time increases. For moderate and large values of *blocksize*, the running time seems to be $O(-1/\log(delta))$ when *delta* is sufficiently large. Moreover, though it has already been mentioned, one can more clearly observe that low and high end *delta* values cause a significant increase in the running time as *blocksize* values get larger or in higher n. Therefore, it is natural to state that BKZ_XD might still be practical for higher *blocksize* values such as $blocksize > 30$ in relatively higher dimensions, if *delta* values are chosen moderately.

As for the achieved SVP approximation factor constants, Figures C.7 through C.12 provide some insight.

In the experiments, no general pattern emerges concerning the behaviour of achieved a values with respect to the parameter n. However, it is observed relaxation of *delta* causes apparent fluctuations and the a vs. n curves seem to be relatively more stable, and increasing proportional to n, for larger values of *delta*. It is noted that more experiments are likely to smoothen the curves, which turn out to be linear, for moderate and large values of *delta*. Furthermore, the cause of the increase in the achieved constants can be stated as follows. Due to the nature of BKZ_XD, the growth of the calculated $a^n = \left(\|\mathbf{b}_1\| / vol(L)^{1/n} \sqrt{2\pi e/n} \right)$ values is high enough that taking the n-th root to obtain the SVP approximation factor constants is not sufficient to cause a decline with the increasing n. As usual, selecting $\log B$ to be fixed and using knapsack lattices (may) all have certain effects on this occurrence.

The effect of *blocksize* parameter on a is given in Figures C.9 and C.10. Since the *blocksize* controls the quality of the computed bases, the value of a decreases with the increasing *blocksize*. This directly leads to decreasing curves in the figure. As in the

previous discussion, increasing the number of experiments smoothen the curves to be more linear.

The *delta* parameter also affects the constants achieved by BKZ_XD. One may obtain better a values by increasing the value of *delta* , since computed $\|\mathbf{b}_1\|$ values have a tendency to decrease. This leads to the graphs shown in Figures C.11 and C.12.

The outcomes of the experiments show that one can consider the following remarks as means to improve the runtime or the achieved approximation factor constants of BKZ_XD reduction.

- Since BKZ_XD seems to be very impractical for $blocksize \geqslant 30$especially for large values of *delta*, it might be possible to achieve equivalently good constants in reasonable time by keeping the chosen *delta* values moderately large and increasing the value of the *blocksize* parameter beyond 30.

- If one manages to find a way to reduce the dimension of the lattice while keeping the lattice properties, such as the type of the lattice and the size parameter $\log B$ etc., (relatively) intact, the achieved approximation factor constants can be improved as well as the runtime performance.

- As noted already, it might be possible to use these type of strategies in combination with each other.

8.1.4. A Short Comparison

A comparison of the running times and achieved approximation factor constants of LLL_XD, LLL_XD_DEEP with $deep = 10, 30, 50$ and BKZ_XD with $blocksize = 10, 20, 30$, in fixed dimension $n = 100$, are given in Figure D.1 and D.2 respectively.

Figure D.1 shows that LLL_XD_DEEP and BKZ_XD for small *delta* and *blocksize* values are much closer to LLL_XD in terms of runtime performance. Furthermore, the impracticality of BKZ_XD with *blocksize* shows that one may prefer to use LLL_XD_DEEP, since it is still possible to increase the *deep* parameter beyond 50 to obtain better obtain a reasonable runtime performance while achieving better a values and higher dimensions.

It can be seen from Figure D.2(b) that as *deep* and *blocksize* parameters are in-

creased, both LLL_XD_DEEP and BKZ_XD achieve similar approximation factor constants. This gives even more reason to prefer LLL_XD_DEEP over BKZ_XD. However, one should keep in mind the reduction strategies noted in the previous sections, which include the relaxation of $delta$ values and increasing $deep$ and $blocksize$ parameters for LLL_XD_DEEP and BKZ_XD respectively, can still be useful to increase the performance of both reduction mechanisms.

8.2. Future Studies

Throughout this study, the most famous lattice basis reduction algorithms are examined in terms of theoretical and practical perspectives while considering their cryptanalytical aspects. The results of the experiments demonstrated in the previous section yield some preliminary practical assessments on the effects of the common $delta$ parameter on the most famous and efficient variants of the LLL and BKZ basis reduction algorithms. It should be noted that, the experiments are performed upon random knapsack lattices where $\log B$ is fixed and the outcomes arouse a couple of questions.

First, it might be convenient to find out whether the decrease in the achieved approximation factor constants for LLL_XD and LLL_XD_DEEP as the dimension of the lattices increases stems only from the fact that $\log B$ is fixed and the lattice are of knapsack type. It might be fruitful to research whether there are other types of lattices which display similar behaviour or whether this occurrence is exploitable in any other way.

Next, it is noted that in sufficiently low dimensions the approximation factor constants achieved by LLL_XD_DEEP surpasses some of the constants achieved in higher dimensions. It is possible that this happens due to the fact that, with respect to the chosen values, the lattice dimension n is too low and thus unexpectedly good values are obtained. Nevertheless, before coming to a strict conclusion, one should investigate whether this experimental occurrence is recurring in much higher dimensions or in other type of lattices.

Also, the runtime behaviour of LLL_XD_DEEP on random knapsack lattices points out that the knapsack lattices may be easier to reduce on the average with respect to other types of lattices. It might be useful to classify the lattices in such a way

that different reduction strategies can be followed for different lattice classes.

Finally, the experiments show that the effect of *delta* on the runtime performance and the output quality of the algorithms, especially LLL_XD_DEEP and BKZ_XD, is significant. When careful measures are taken, these algorithms may yield better results running with even higher *deep* and *blocksize* parameters in higher dimensions.

In the light of the above remarks, it seems to be useful to extend the discussion to other types of lattices and perhaps to also investigate the distribution of $\|\mathbf{b}_1\|$ and $vol\,(L)^{1/n}$ values and the Gram-Schmidt coefficients etc. in order to obtain better idea on the output quality of the considered algorithms. The possible criterion and background which might be helpful to perform such a research can be found in (Backes and Wetzel 2002, Nguyen and Stehlé 2006, Gama and Nguyen 2008b).

REFERENCES

Aardal, K. I. 1999. Lattice basis reduction and integer programming. Technical Report UU-CS-1999-37, Utrecht University.

Agrell, E., T. Eriksson, A. Vardy, and K. Zeger. 2002. Closest point search in lattices. *IEEE Transactions on Information Theory 48*(8): 2201–2214.

Aharonov, D. and O. Regev. 2003. A lattice problem in quantum NP. *FOCS '03: Proceedings of the 44th Annual IEEE Symposium on Foundations of Computer Science* 210–219.

Aharonov, D. and O. Regev. 2005. Lattice problems in NP ∩ coNP. *Journal of the ACM 52*(5): 749–765.

Ajtai, M. 1996. Generating hard instances of lattice problems (extended abstract). *STOC '96: Proceedings of the 28th Annual ACM Symposium on Theory of Computing* 99–108.

Ajtai, M. 1998. The shortest vector problem in l_2 is NP-hard for randomized reductions (extended abstract). *STOC '98: Proceedings of the 30th Annual ACM Symposium on Theory of Computing* 10–19.

Ajtai, M. 2002. Random lattices and a conjectured 0 - 1 law about their polynomial time computable properties. *FOCS '02: Proceedings of the 43rd Annual IEEE Symposium on Foundations of Computer Science* 733–742.

Ajtai, M. 2003. The worst-case behavior of schnorr's algorithm approximating the shortest nonzero vector in a lattice. *STOC '03: Proceedings of the 35th Annual ACM Symposium on Theory of Computing* 396–406.

Ajtai, M. 2005. Representing hard lattices with o(n log n) bits. *STOC '05: Proceedings of the 37th Annual ACM Symposium on Theory of Computing* 94–103.

Ajtai, M. and C. Dwork. 1997. A public-key cryptosystem with worst-case/average-case equivalence. *STOC '97: Proceedings of the 29th Annual ACM Symposium on Theory of Computing* 284–293.

Ajtai, M., R. Kumar, and D. Sivakumar. 2001. A sieve algorithm for the shortest lattice vector problem. *STOC '01: Proceedings of the 33rd Annual ACM Symposium on Theory of Computing* 601–610.

Ajtai, M., R. Kumar, and D. Sivakumar. 2002. Sampling short lattice vectors and the closest lattice vector problem. *COCO '02: Proceedings of the 17th Annual IEEE Conference on Computational Complexity* 41–45.

Akhavi, A. 2002. Random lattices, threshold phenomena and efficient reduction algorithms. *Theoretical Computer Science 287*(2): 359–385.

Akhavi, A. 2003. The optimal LLL algorithm is still polynomial in fixed dimension. *Theoretical Computer Science 297*(1-3): 3–23.

Akhavi, A. and D. Stehlé. 2008. Speeding-up lattice reduction with random projections (extended abstract). *LATIN 2008: Theoretical Informatics* 4957/2008 of *Lecture Notes in Computer Science* 293–305.

Arora, S., L. Babai, J. Stern, and Z. Sweedyk. 1997. The hardness of approximate optima in lattices, codes, and systems of linear equations. *Journal of Computer and System Sciences 54*(2): 317–331.

Babai, L. 1986. On Lovász' lattice reduction and the nearest lattice point problem. *Combinatorica 6*(1): 1–13.

Backes, W. and S. Wetzel. 2001. Lattice basis reduction with dynamic approximation. *WAE '00: Proceedings of the 4th International Workshop on Algorithm Engineering* 1982/2001: 63–73.

Backes, W. and S. Wetzel. 2002. Heuristics on lattice basis reduction in practice. *Journal of Experimental Algorithmics 7* 1–21.

Banaszczyk, W. 1993. New bounds in some transference theorems in the geometry of numbers. *Mathematische Annalen 296*(1): 625–635.

Blömer, J. 2000. Closest vectors, successive minima, and dual HKZ-Bases of lattices. *Automata, Languages and Programming* 1853/2000: 248–259.

Blömer, J. and S. Naewe. 2007. Sampling methods for shortest vectors, closest vectors and successive minima. *Automata, Languages and Programming* 4596/2007: 65–77.

Blömer, J. and J.-P. Seifert. 1999. On the complexity of computing short linearly independent vectors and short bases in a lattice. *STOC '99: Proceedings of the 31st Annual ACM Symposium on Theory of Computing* 711–720.

Buchmann, J. and V. Kessler. 1990. Computing a reduced lattice basis from a generating system with applications.

Buchmann, J. and C. Ludwig. 2006. Practical lattice basis sampling reduction. *Algorithmic Number Theory* 4076/2006: 222–237.

Cai, J.-Y. 1998. A relation of primal-dual lattices and the complexity of shortest lattice vector problem. *Theoretical Computer Science 207*(1): 105–116.

Cai, J.-Y. 1999. Some recent progress on the complexity of lattice problems. *COCO '99: Proceedings of the 14th Annual IEEE Conference on Computational Complexity* 158–178.

Cai, J.-Y. 2000. The complexity of some lattice problems. *Algorithmic Number Theory* 1838/2000: 1–32.

Cai, J.-Y. 2001. On the average-case hardness of CVP. *FOCS '01: Proceedings of the 42nd IEEE Symposium on Foundations of Computer Science* 308–317.

Cai, J.-Y. and T. W. Cusick. 1999. A lattice-based public-key cryptosystem. *Information and Computation 151*(1-2): 17–31.

Cai, J.-Y. and A. Nerurkar. 1997. An improved worst-case to average-case connection for lattice problems. *FOCS '97: Proceedings of the 38th Annual IEEE Symposium on Foundations of Computer Science* 468–477.

Cai, J.-Y. and A. Nerurkar. 1999. Approximating the SVP to within a factor $1 + 1/dim^{\varepsilon}$ is NP-Hard under randomized reductions. *Journal of Computer and System Sciences 59*(2): 221–239.

Cai, J.-Y. and A. Nerurkar. 2000. A note on the non-NP-hardness of approximate lattice problems under general Cook reductions. *Information Processing Letters 76*(1-2): 61–66.

Cassels, J. 1997. *An introduction to the geometry of numbers*. Berlin: Springer-Verlag.

Chen, W. and J. Meng. 2006. The hardness of the closest vector problem with preprocessing over l_∞ norm. *IEEE Transactions on Information Theory 52*(10): 4603–4606.

Coppel, W. A. 2006. The geometry of numbers. In *Number theory*, 385–426. Springer US.

Coppersmith, D. and A. Shamir. 1997. Lattice attacks on NTRU. *Advances in Cryptology - EUROCRYPT '97* 1233/1997: 52–61.

Daudé, H. and B. Vallée. 1994. An upper bound on the average number of iterations of the LLL algorithm. *Theoretical Computer Science 123*(1): 95–115.

Dinur, I. 2000. Approximating SVP_∞ within almost-polynomial factors is NP-Hard. *Algorithms and Complexity* 1767/2000: 263–276.

Dinur, I. 2002. Approximating SVP_∞ to within almost-polynomial factors is NP-hard. *Theoretical Computer Science 285*(1): 55–71.

Dinur, I., G. Kindler, R. Raz, and S. Safra. 2003. Approximating CVP to within almost-polynomial factors is NP-Hard. *Combinatorica 23*(2): 205–243.

Dinur, I., G. Kindler, and S. Safra. 1998. Approximating-CVP to within almost-polynomial factors is NP-hard. *FOCS '98: Proceedings of the 39th Annual IEEE Symposium on Foundations of Computer Science* 99–109.

Dwork, C. 1998. Lattices and their application to cryptography. University of California. http://www.math.ucdavis.edu/ %7Edeloera/misc/mathresearch/interesting_papers/Dwork/gitter.ps (accessed at April 14, 2008).

Fischlin, R. and J.-P. Seifert. 1999. Tensor-based trapdoors for CVP and their application to public key cryptography (extended abstract). *Cryptography and Coding* 1746/1999: 801–814.

Furst, M. L. and R. Kannan. 1989. Succinct certificates for almost all subset sum problems. *SIAM Journal on Computing 18*(3): 550–558.

Gama, N., N. Howgrave-Graham, H. Koy, and P. Q. Nguyen. 2006. Rankin's constant and blockwise lattice reduction. *Advances in Cryptology - CRYPTO 2006* 4117/2006: 112–130.

Gama, N., N. Howgrave-Graham, and P. Q. Nguyen. 2006. Symplectic lattice reduction and NTRU. *Advances in Cryptology - EUROCRYPT 2006* 4004/2006: 233–253.

Gama, N. and P. Q. Nguyen. 2008a. Finding short lattice vectors within Mordell's inequality. *STOC '08: Proceedings of the 40th ACM Symposium on the Theory of Computing*.

Gama, N. and P. Q. Nguyen. 2008b. Predicting lattice reduction. *Advances in Cryptology – Proc. Eurocrypt '08*.

Garey, M. R. and D. S. Johnson. 1990. *Computers and intractability; a guide to the theory of NP-completeness*. New York: W. H. Freeman & Co.

Gentry, C. 2001. Key recovery and message attacks on NTRU-Composite. *Advances in Cryptology - EUROCRYPT 2001* 2045/2001: 182–194.

Gentry, C., C. Peikert, and V. Vaikuntanathan. 2007. Trapdoors for hard lattices and new cryptographic constructions. Cryptology ePrint Archive, Report 2007/432. http://eprint.iacr.org/ (accessed at May 13, 2008).

Gentry, C. and M. Szydlo. 2002. Cryptanalysis of the revised NTRU signature scheme. *Advances in Cryptology - EUROCRYPT 2002* 2332/2002: 299–320.

Goldreich, O. and S. Goldwasser. 2000. On the limits of nonapproximability of lattice problems. *Journal of Computer and System Sciences 60*(3): 540–563.

Goldreich, O., S. Goldwasser, and S. Halevi. 1997. Public-key cryptosystems from lattice reduction problems. *CRYPTO '97: Proceedings of the 17th Annual International Cryptology Conference on Advances in Cryptology* 1294/1997: 112–131.

Goldreich, O., D. Micciancio, S. Safra, and J.-P. Seifert. 1999. Approximating shortest lattice vectors is not harder than approximating closest lattice vectors. *Information Processing Letters 71*(2): 55–61.

Goldstein, D. and A. Mayer. 2003. On the equidistribution of hecke points. *Forum Mathematicum 15*(2): 165–189.

Gruber, P. M. and C. G. Lekkerkerker. 1987. *Geometry of numbers*. Amsterdam: North-Holland.

Guruswami, V., D. Micciancio, and O. Regev. 2005. The complexity of the covering radius problem. *Computational Complexity 14*(2): 90–121.

Han, D., M.-H. Kim, and Y. Yeom. 2007. Cryptanalysis of the Paeng-Jung-Ha cryptosystem from PKC 2003. *Public Key Cryptography - PKC 2007* 4450/2007: 107–117.

Hanrot, G. and D. Stehlé. 2007. Improved analysis of Kannan's shortest lattice vector algorithm. *Advances in Cryptology - CRYPTO 2007* 4622/2007: 170–186.

Hanrot, G. and D. Stehlé. 2008. Worst-case Hermite-Korkine-Zolotarev reduced lattice bases. *Computing Research Repository abs/0801.3331*. http://arxiv.org/abs/0801.3331 (accessed at April 25, 2008).

Hastad, J. 1988. Dual vectors and lower bounds for the nearest lattice point problem. *Combinatorica 8*(1): 75–81.

Haviv, I. and O. Regev. 2006. Hardness of the covering radius problem on lattices. *COCO '06: Proceedings of the 21st Annual IEEE Conference on Computational Complexity* 145–158.

Haviv, I. and O. Regev. 2007. Tensor-based hardness of the shortest vector problem to within almost polynomial factors. *STOC '07: Proceedings of the 39th Annual ACM Symposium on Theory of Computing* 469–477.

Heckler, C. and L. Thiele. 1993a. Parallel complexity of lattice basis reduction and a floating-point parallel algorithm. *PARLE '93 Parallel Architectures and Languages Europe* 694/1993: 744–747.

Heckler, C. and L. Thiele. 1993b. A parallel lattice basis reduction for mesh-connected processor arrays and parallel complexity. *Proceedings of the 5th IEEE Symposium on Parallel and Distributed Processing* 400–407.

Heckler, C. and L. Thiele. 1998. Complexity analysis of a parallel lattice basis reduction algorithm. *SIAM Journal on Computing 27*(5): 1295–1302.

Helfrich, B. 1985. Algorithms to construct Minkowski reduced and Hermite reduced lattice bases. *Theoretical Computer Science 41*(2-3): 125–139.

Henk, M. 1997. Note on shortest and nearest lattice vectors. *Information Processing Letters 61*(4): 183–188.

Hoffstein, J., D. Lieman, J. Pipher, and J. H. Silverman. 1999. NTRU: A public key cryptosystem. http://grouper.ieee.org/groups/1363/lattPK/submissions.html (accessed at April 27, 2008).

Hoffstein, J., J. Pipher, and J. H. Silverman. 1998. NTRU: A ring-based public key cryptosystem. *Algorithmic Number Theory* 1423/1998: 267–288.

Joux, A. and J. Stern. 1998. Lattice reduction: A toolbox for the cryptanalyst. *Journal of Cryptology 11*(3): 161–185.

Kaib, M. and H. Ritter. 1994. Block reduction for arbitrary norms. Technical report, Goethe Universität. http://www.mi.informatik.uni-frankfurt.de/research/papers.html (accessed at April 25, 2008).

Kaltofen, E. 1983. On the complexity of finding short vectors in integer lattices. *Computer Algebra* 162/1983: 236–244.

Kannan, R. 1983. Improved algorithms for integer programming and related lattice problems. *STOC '83: Proceedings of the 15th Annual ACM Symposium on Theory of Computing* 193–206.

Kannan, R. 1987a. Algorithmic geometry of numbers. *Annual Review of Computer Science 2*(1): 231–267.

Kannan, R. 1987b. Minkowski's convex body theorem and integer programming. *Mathematics of Operations Research 12*(3): 415–440.

Kawachi, A., K. Tanaka, and K. Xagawa. 2007. Multi-bit cryptosystems based on lattice problems. *Public Key Cryptography - PKC 2007* 4450/2007: 315–329.

Khot, S. 2003. Hardness of approximating the shortest vector problem in high l_p norms. *FOCS '03: Proceedings of the 44th Annual IEEE Symposium on Foundations of Computer Science* 290–303.

Khot, S. 2005. Hardness of approximating the shortest vector problem in lattices. *Journal of the ACM 52*(5): 789–808.

Klein, P. 2000. Finding the closest lattice vector when it's unusually close. *SODA '00: Proceedings of the 11th Annual ACM-SIAM Symposium on Discrete Algorithms* 937–941.

Koy, H. 2004. Primale duale segment-reduktion von gitterbasen. Goethe Universität. http://www.mi.informatik.uni-frankfurt.de/research/papers.html (accessed at April 25, 2008).

Koy, H. and C. P. Schnorr. 2001a. Segment LLL-reduction of lattice bases. *Cryptography and Lattices* 2146/2001: 67–80.

Koy, H. and C. P. Schnorr. 2001b. Segment LLL-reduction with floating point orthogonalization. *Cryptography and Lattices* 2146/2001: 81–96.

Koy, H. and C. P. Schnorr. 2002. Segment and strong segment LLL-reduction of lattice bases. Technical report, Goethe Universität. http://www.mi.informatik.uni-frankfurt.de/research/papers.html (accessed at April 25, 2008).

Kumar, R. and D. Sivakumar. 1999. A note on the shortest lattice vector problem. *COCO '99: Proceedings of the 14th Annual IEEE Conference on Computational Complexity* 200–204.

Kumar, R. and D. Sivakumar. 2001. On polynomial approximation to the shortest lattice vector length. *SODA '01: Proceedings of the 12th Annual ACM-SIAM Symposium on Discrete Algorithms* 126–127.

Lagarias, J. C. 1995. Point lattices. In *Handbook of combinatorics 1*, 919–966. Cambridge: MIT Press.

Lagarias, J. C., H. W. L. Jr., and C. P. Schnorr. 1990. Korkin-Zolotarev bases and successive minima of a lattice and its reciprocal lattice. *Combinatorica 10*(4): 333–348.

Lenstra, A. K. 1981. Lattices and factorization of polynomials. *SIGSAM Bull. 15*(3): 15–16.

Lenstra, A. K., H. W. L. Jr., and L. Lovász. 1982. Factoring polynomials with rational coefficients. *Mathematische Annalen 261*(4): 515–534.

Lovász, L. and H. E. Scarf. 1992. The generalized basis reduction algorithm. *Mathematics of Operations Research 17*(3): 751–764.

Ludwig, C. 2005. *Practical lattice basis sampling reduction*. PhD Diss., Fachbereich Informatik, TU Darmstadt.

May, A. 1999. Cryptanalysis of NTRU. http://citeseer.ist.psu.edu/may99cryptanalysis.html (accessed at April 27,2008).

May, A. and J. H. Silverman. 2001. Dimension reduction methods for convolution modular lattices. *Cryptography and Lattices* 2146/2001: 110–125.

Micciancio, D. 1998. *On the hardness of the shortest vector problem*. PhD Diss., Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology.

Micciancio, D. 1999. Lattice based cryptography: A global improvement. Cryptology ePrint Archive, Report 1999/005. http://eprint.iacr.org/ (accessed at April 1, 2008).

Micciancio, D. 2001a. Improving lattice based cryptosystems using the Hermite normal form. *Cryptography and Lattices* 2146/2001: 126–145.

Micciancio, D. 2001b. The shortest vector in a lattice is hard to approximate to within some constant. *SIAM Journal on Computing 30*(6): 2008–2035.

Micciancio, D. 2002. Lattices in cryptography and cryptanalysis. UC San Diego. http://www-cse.ucsd.edu/%7Edaniele/CSE207C/ (accessed at April 14, 2008).

Micciancio, D. 2007. Cryptographic functions from worst-case complexity assumptions. *LLL +25*. UC San Diego. http://www-cse.ucsd.edu/%7Edaniele/index.html (accessed at June 2, 2008).

Micciancio, D. 2008. Efficient reductions among lattice problems. *SODA '08: Proceedings of the 19th Annual ACM-SIAM Symposium on Discrete Algorithms* 84–93.

Micciancio, D. and S. Goldwasser. 2002. *Complexity of lattice problems: a cryptographic perspective*. Boston: Kluwer Academic Publishers.

Nguyen, P. 1999. Cryptanalysis of the Goldreich-Goldwasser-Halevi cryptosystem from crypto '97. *Advances in Cryptology - CRYPTO' 99* 1666/1999: 790–806.

Nguyen, P. and J. Stern. 1998. Cryptanalysis of the Ajtai-Dwork cryptosystem. *Advances in Cryptology - CRYPTO '98* 1462/1998: 223–242.

Nguyen, P. Q. 2001. The two faces of lattices in cryptology. *Selected Areas in Cryptography* 2259/2001: 313–347.

Nguyen, P. Q. 2008. Public-key cryptanalysis. In *Recent Trends in Cryptography*. AMS–RSME.

Nguyen, P. Q. and O. Regev. 2006. Learning a parallelepiped: Cryptanalysis of GGH and NTRU signatures. *Advances in Cryptology - EUROCRYPT 2006* 4004/2006: 271–288.

Nguyen, P. Q. and D. Stehlé. 2006. LLL on the average. *Algorithmic Number Theory* 4076/2006: 238–256.

Nguyen, P. Q. and D. Stehlé. 2007. An algorithm with quadratic complexity. Updated version of P. Q. Nguyen and D. Stehlés. 2005. Floating-Point LLL Revisited. Advances in Cryptology - Proceedings of EUROCRYPT 05 3494: 215-233. http://www.di.ens.fr/%7Epnguyen/pub.html (accessed at April 25, 2008).

Nguyen, P. Q. and T. Vidick. 2008. Sieve algorithms for the shortest vector problem are practical. *Journal of Mathematical Cryptology*. ENS. http://www.di.ens.fr/%7Epnguyen/pub.html (accessed at April 25, 2008).

Paeng, S.-H., B. E. Jung, and K.-C. Ha. 2002. A lattice based public key cryptosystem using polynomial representations. *Public Key Cryptography - PKC 2003* 2567/2002: 292–308.

Peikert, C. 2007. Limits on the hardness of lattice problems in l_p norms. *COCO '07: Proceedings of the 22nd Annual IEEE Conference on Computational Complexity* 333–346.

Peikert, C. and B. Waters. 2007. Lossy trapdoor functions and their applications. Cryptology ePrint Archive, Report 2007/279. http://eprint.iacr.org/ (accessed at May 13, 2008).

Pendavingh, R. A. and S. H. M. van Zwam. 2007. New Korkin-Zolotarev inequalities. *SIAM Journal on Optimization 18*(1): 364–378.

Pohst, M. 1987. A modification of the LLL reduction algorithm. *Journal of Symbolic Computation 4*(1): 123–127.

Regev, O. 2004a. Lattices in computer science. Tel-Aviv University. http://www.cs.tau.ac.il/%7Eodedr/teaching/lattices_fall_2004/index.html (accessed at April 14, 2008).

Regev, O. 2004b. New lattice-based cryptographic constructions. *Journal of the ACM 51*(6): 899–942.

Regev, O. 2004c. Quantum computation and lattice problems. *SIAM Journal on Computing 33*(3): 738–760.

Regev, O. 2005. On lattices, learning with errors, random linear codes, and cryptography. *STOC '05: Proceedings of the 37th Annual ACM Symposium on Theory of Computing* 84–93.

Regev, O. 2007. On the complexity of lattice problems with polynomial approximation factors. *LLL +25*. Tel-Aviv University. http://www.cs.tau.ac.il/%7Eodedr/ (accessed at March 31, 2008).

Regev, O. and R. Rosen. 2006. Lattice problems and norm embeddings. *STOC '06: Proceedings of the 38th Annual ACM Symposium on Theory of Computing* 447–456.

Roch, J. L. and G. Villard. 1992. Parallel gcd and lattice basis reduction. *Parallel Processing: CONPAR 92-VAPP V* 634/1992: 557–564.

Rothe, J. 2005. *Complexity theory and cryptology*. New York: Springer-Verlag.

Sakurai, K. 2000. A progress report on lattice based public-key cryptosystems : Theoretical security versus practical cryptanalysis(special issue on algorithm engineering : Surveys). *IEICE Transactions on Information and Systems 83*(3): 570–579.

Schnorr, C. P. 1987. A hierarchy of polynomial time lattice basis reduction algorithms. *Theoretical Computer Science 53*(2-3): 201–224.

Schnorr, C. P. 1988. A more efficient algorithm for lattice basis reduction. *Journal of Algorithms 9*(1): 47–62.

Schnorr, C. P. 1996. Block reduced bases and successive minima. Technical report, Goethe Universität. http://www.mi.informatik.uni-frankfurt.de/research/papers.html accessed at April 25, 2008).

Schnorr, C. P. 2001. New practical algorithms for the approximate shortest lattice vector. Goethe Universität. http://www.mi.informatik.uni-frankfurt.de/research/papers.html (accessed at April 25, 2008).

Schnorr, C. P. 2003. Lattice reduction by random sampling and birthday methods. *STACS '03: Proceedings of the 20th Annual Symposium on Theoretical Aspects of Computer Science* 2607/2003: 145–156.

Schnorr, C. P. 2006a. Blockwise lattice basis reduction revisited. *Codes and Lattices in Cryptography*. Goethe Universität. http://www.mi.informatik.uni-frankfurt.de/research/papers.html (accessed at April 25, 2008).

Schnorr, C. P. 2006b. Fast LLL-type lattice reduction. *Information and Computation 204*(1): 1–25.

Schnorr, C. P. 2007. Progress on LLL and lattice reduction. *LLL +25*. Goethe Universität. http://www.mi.informatik.uni-frankfurt.de/research/papers.html (accessed at April 25, 2008).

Schnorr, C. P. and M. Euchner. 1994. Lattice basis reduction: Improved practical algorithms and solving subset sum problems. *Mathematical Programming 66*(1-3): 181–199.

Schnorr, C. P. and H. H. Hörner. 1995. Attacking the Chor-Rivest cryptosystem by improved lattice reduction. *Advances in Cryptology - EUROCRYPT '95* 921/1995: 1–12.

Schönhage, A. 1984. Factorization of univariate integer polynomials by diophantine approximation and an improved basis reduction algorithm. *Automata, Languages and Programming* 172/1984: 436–447.

Siegel, C. L. 1989. *Lectures on the geometry of numbers*. New York: Springer-Verlag.

Silverman, J. H. 2006. An introduction to the theory of lattices. Brown University. http://www.math.brown.edu/%7Ejhs/Presentations/WyomingLattices.pdf (accessed at May 1, 2008).

Stehlé, D. 2007. Floating-point LLL: Theoretical and practical aspects. *LLL +25*. ENS Lyon. http://perso.ens-lyon.fr/damien.stehle/ (accessed at April 25, 2008).

Storjohann, A. 1996. Faster algorithms for integer lattice basis reduction. Technical Report 249, Swiss Federal Institute of Technology.

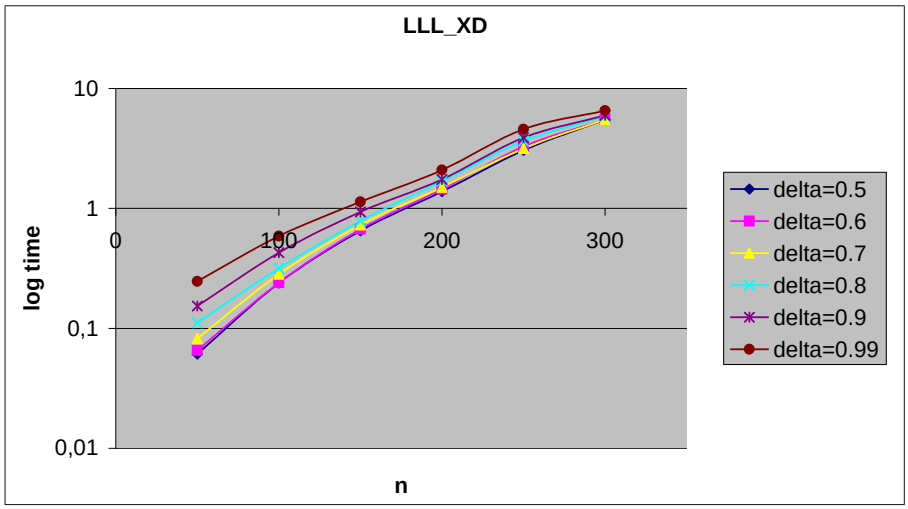
van Emde Boas, P. 1981. Another NP-complete partition problem and the complexity of computing short vectors in a lattice. Technical Report 81-04, University of Amsterdam. http://staff.science.uva.nl/%7Epeter/vectors/mi8104c.html (accessed at March 19, 2008).

Villard, G. 1992. Parallel lattice basis reduction. *ISSAC '92: Proceedşngs of the International 1992 Symposium on Symbolic and Algebraic Computation* 269–277.

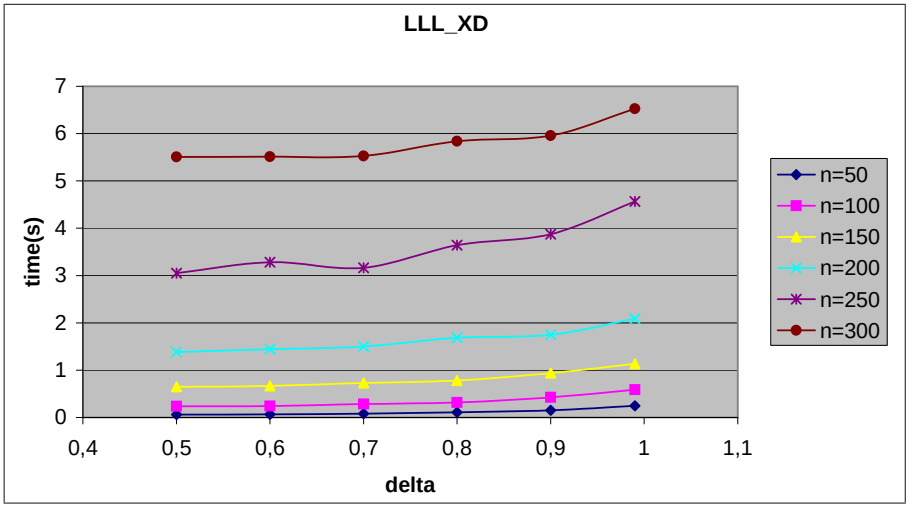
Wetzel, S. 1998. An efficient parallel block-reduction algorithm. *Algorithmic Number Theory* 1423/1998: 323–337.

APPENDIX A

LLL_XD GRAPHS

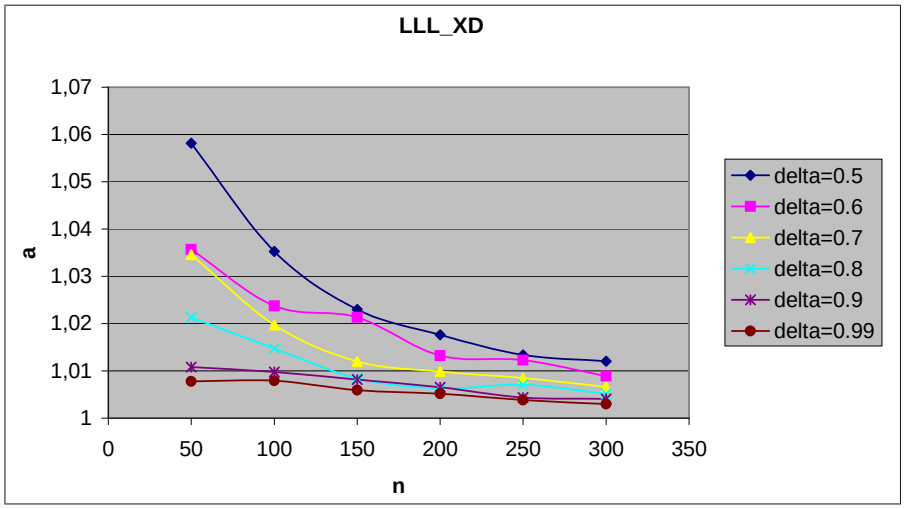


(a) fixed *delta*

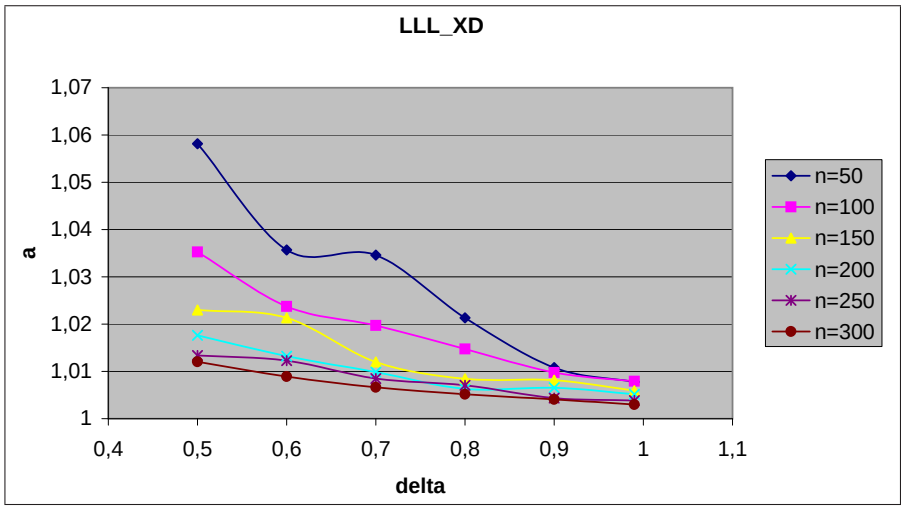


(b) fixed n

Figure A.1. Running times of LLL_XD.



(a) fixed *delta*



(b) fixed n

Figure A.2. Approximation factor constants of LLL_XD.

APPENDIX B

LLL_XD_DEEP GRAPHS

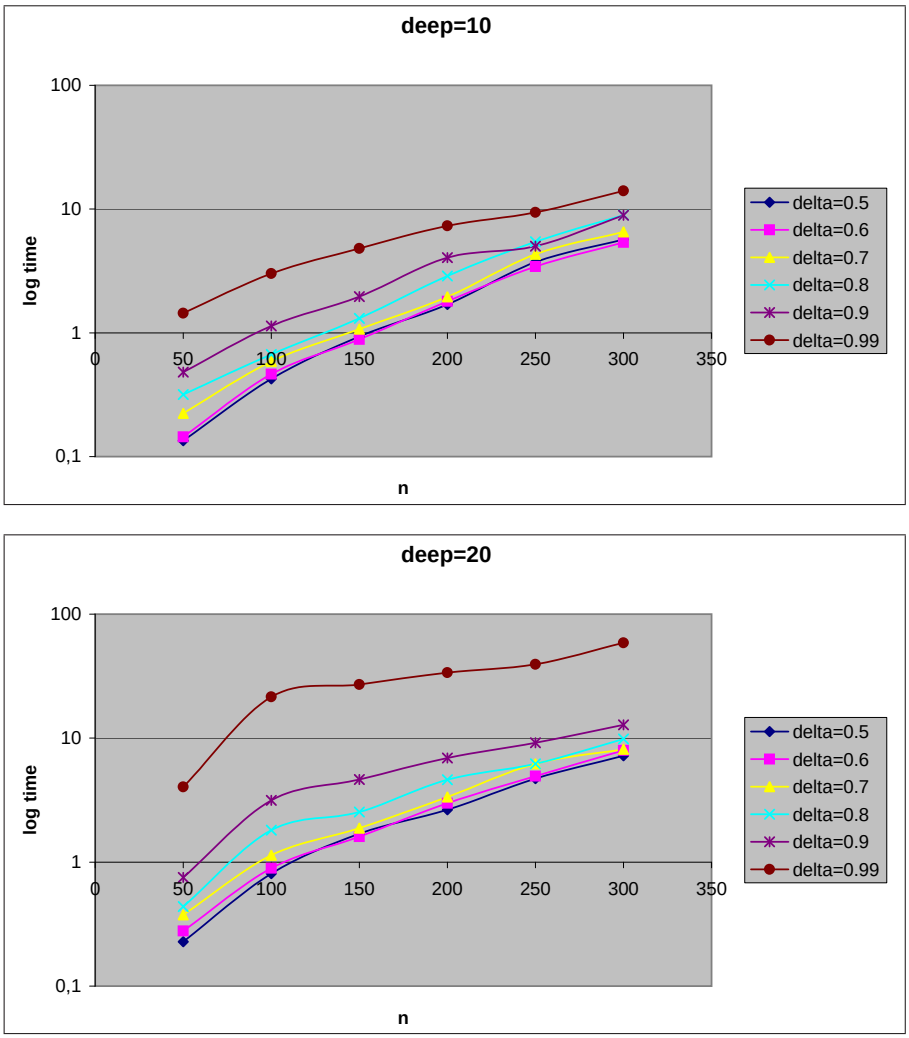


Figure B.1. Running times of LLL_XD_DEEP in *n* (1).

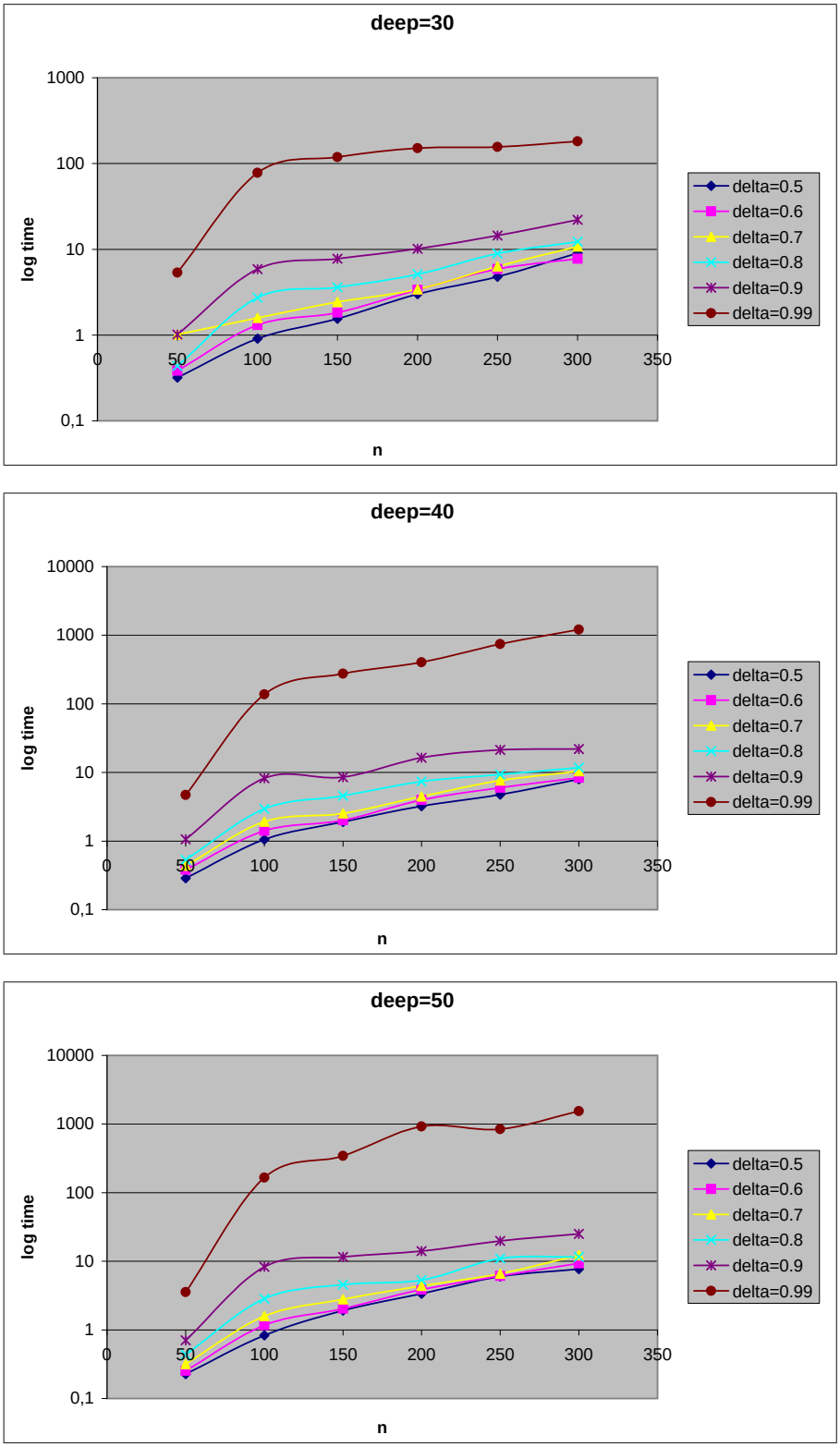


Figure B.2. Running times of LLL_XD_DEEP in n (2).

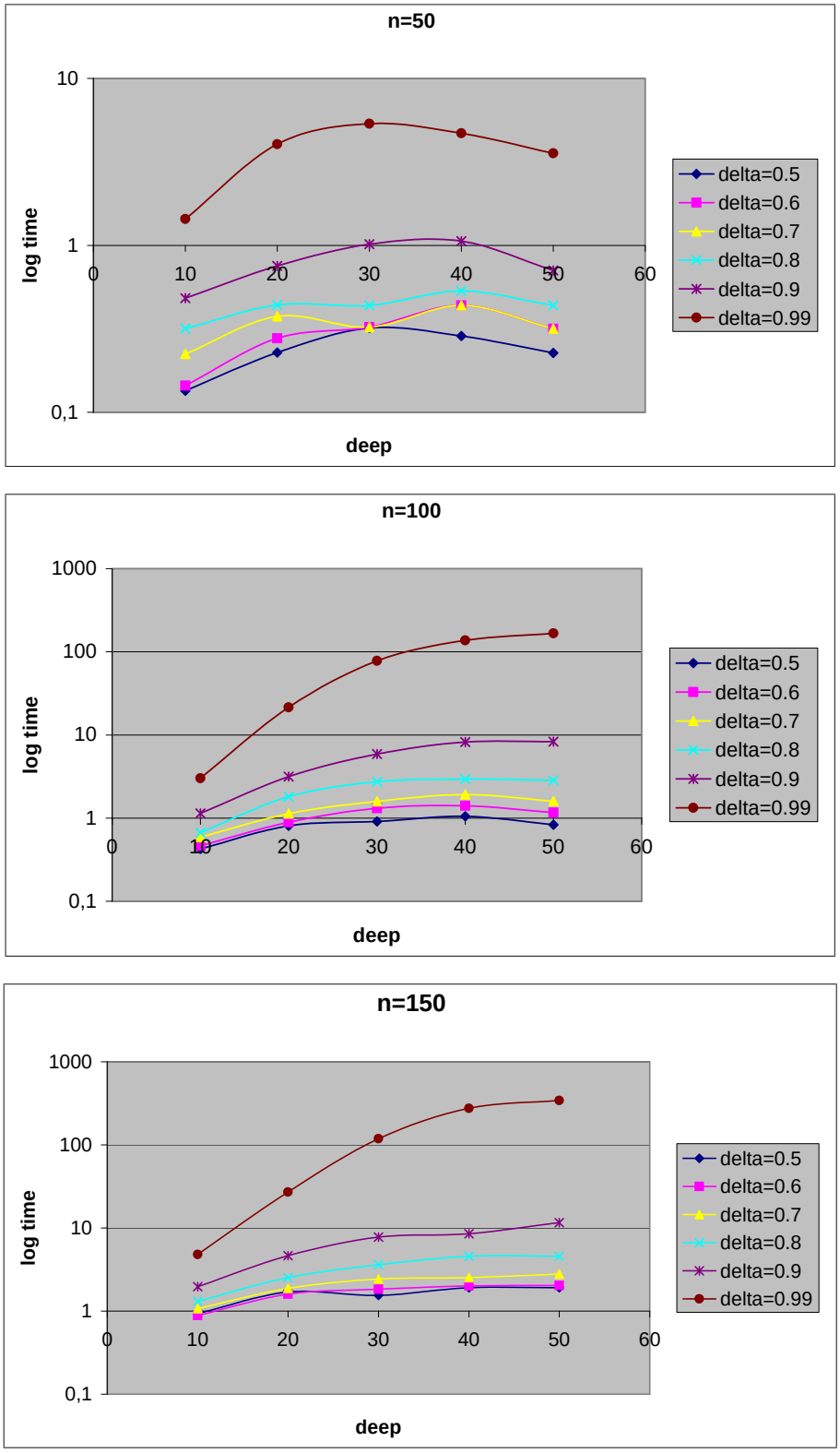


Figure B.3. Running times of LLL_XD_DEEP in *deep* (1).

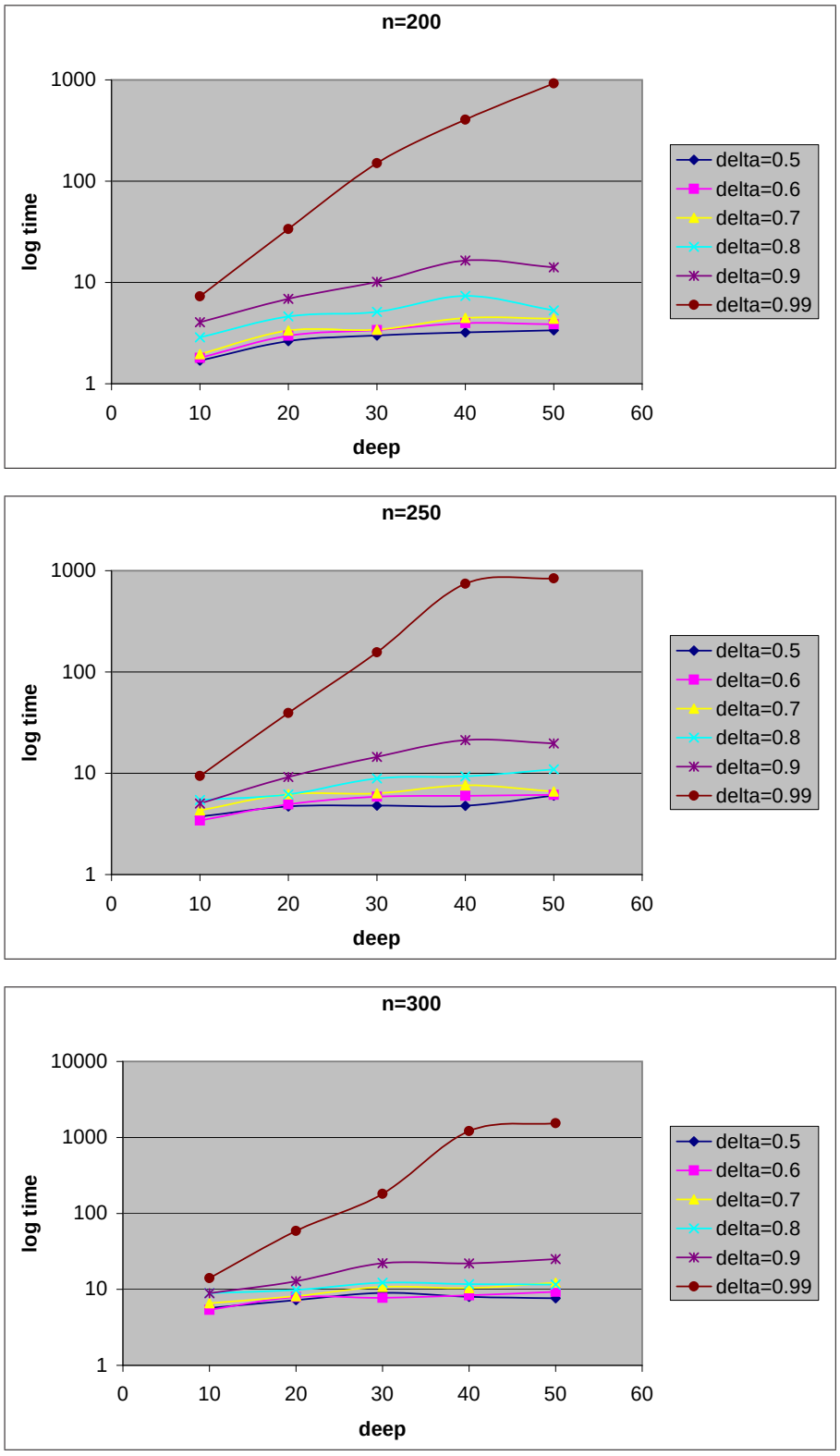


Figure B.4. Running times of LLL_XD_DEEP in *deep* (2).

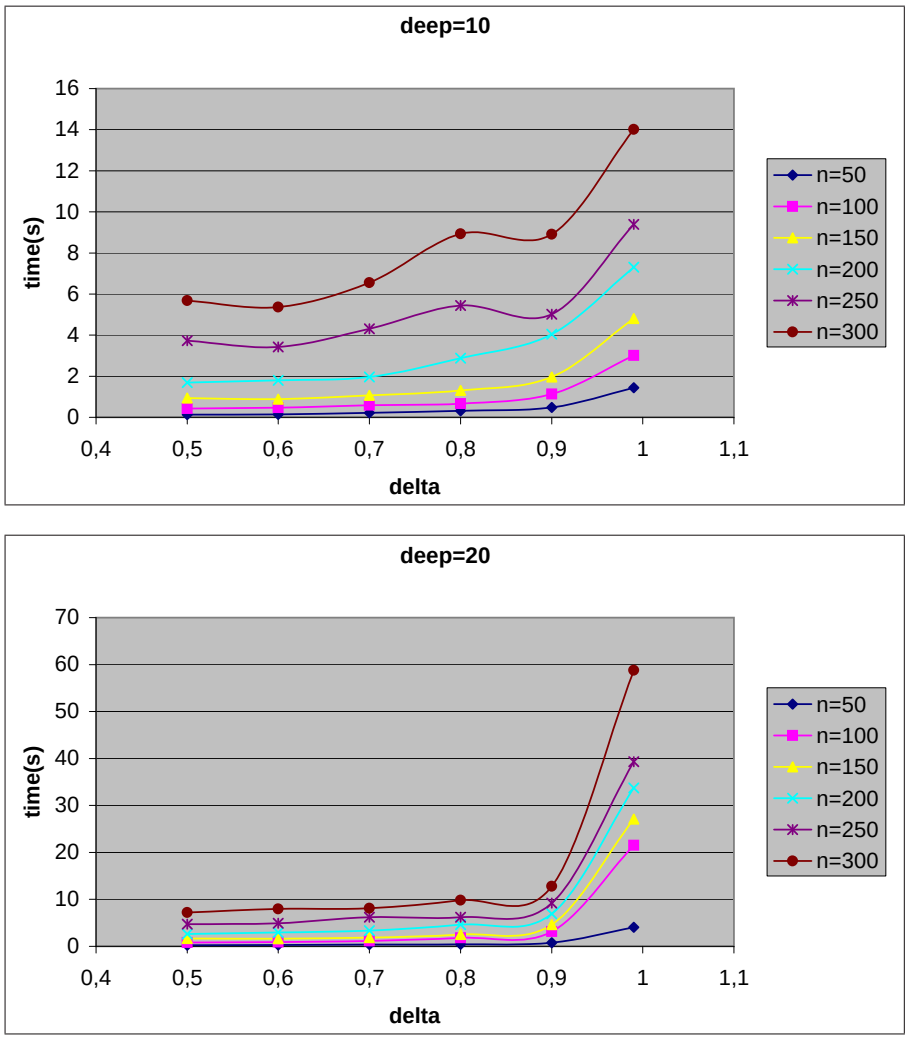


Figure B.5. Running times of LLL_XD_DEEP in *delta* (1).

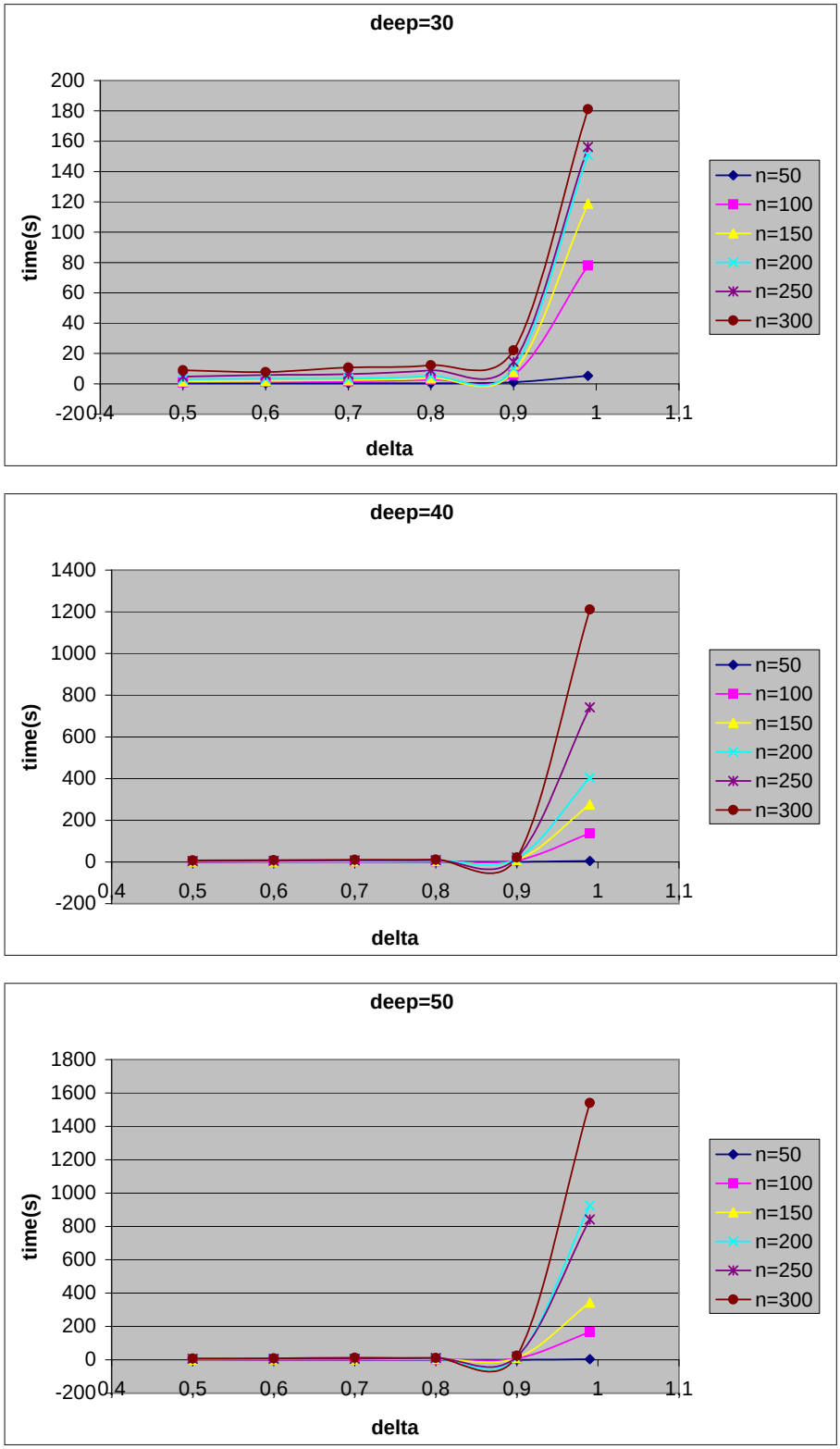


Figure B.6. Running times of LLL_XD_DEEP in *delta* (2).

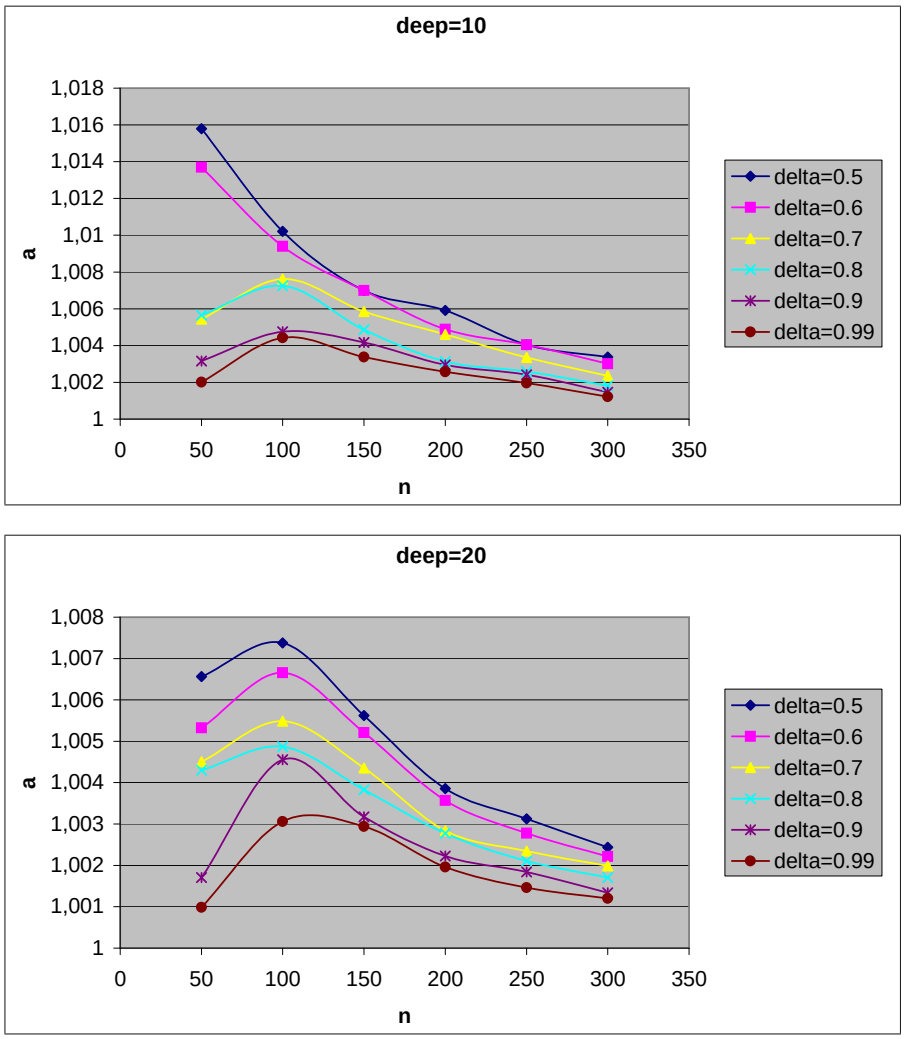


Figure B.7. Approximation factor constants of LLL_XD_DEEP in n (1).

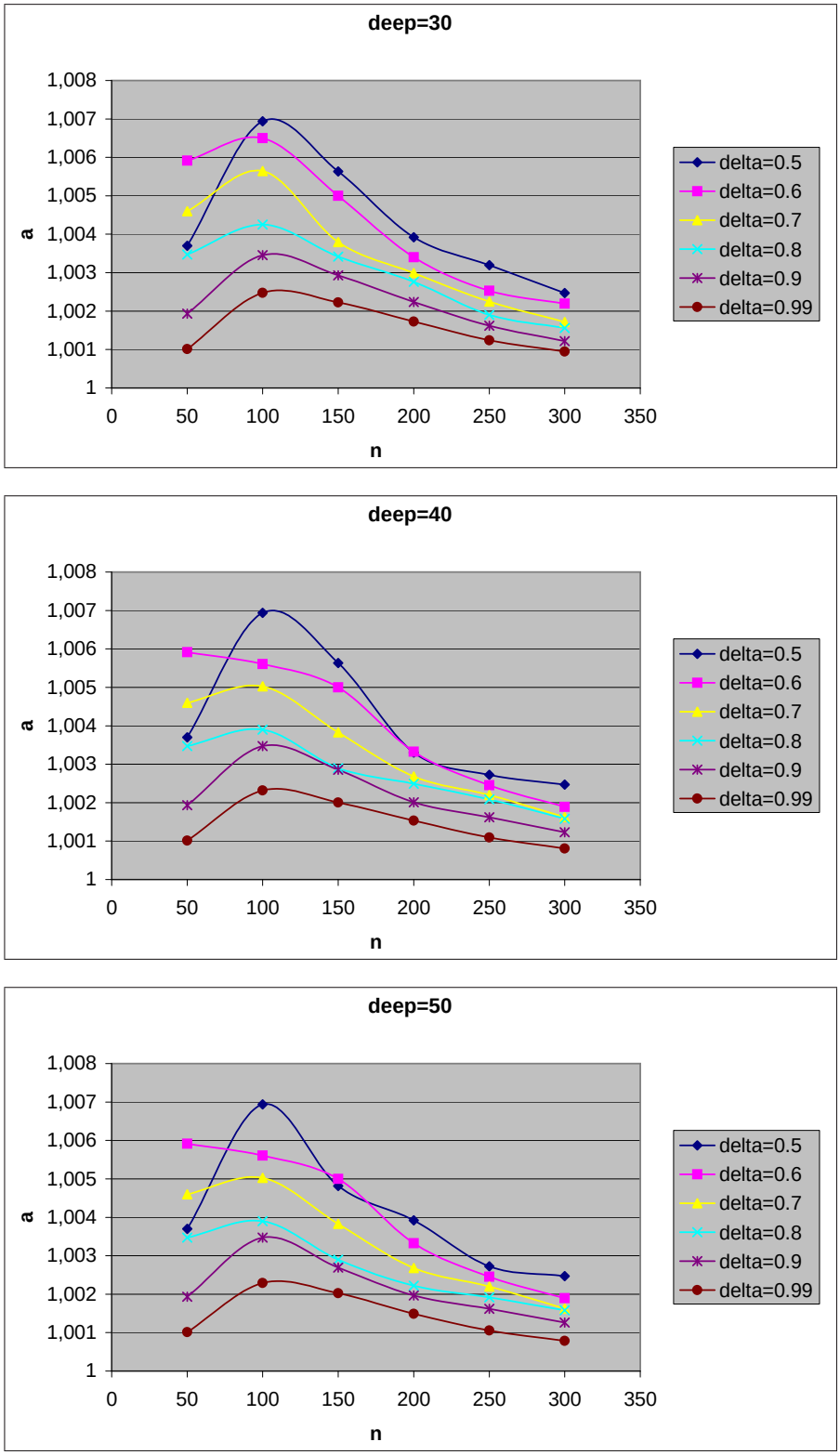


Figure B.8. Approximation factor constants of LLL_XD_DEEP in n (2).

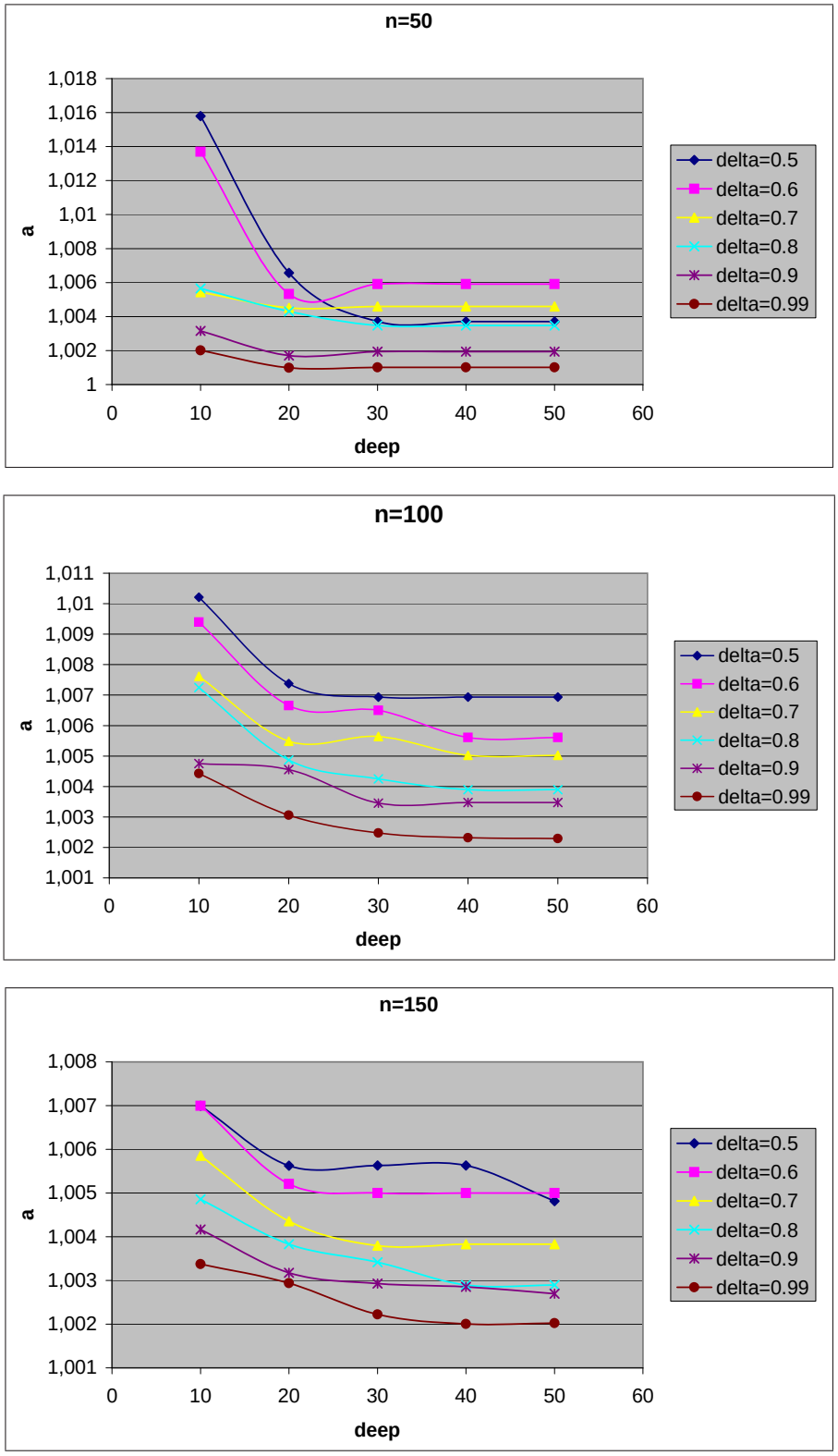


Figure B.9. Approximation factor constants of LLL_XD_DEEP in *deep* (1).

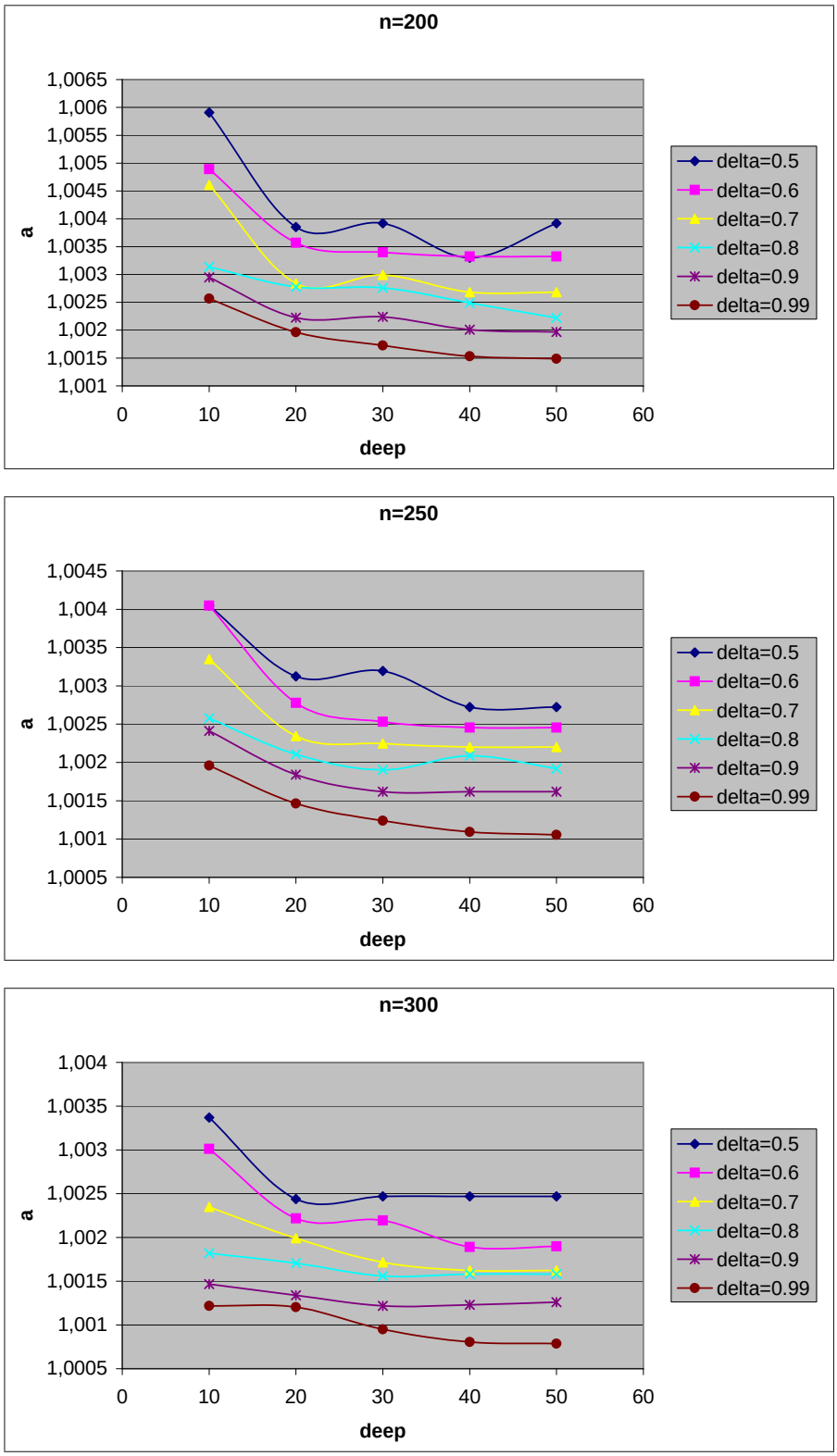


Figure B.10. Approximation factor constants of LLL_XD_DEEP in *deep* (2).

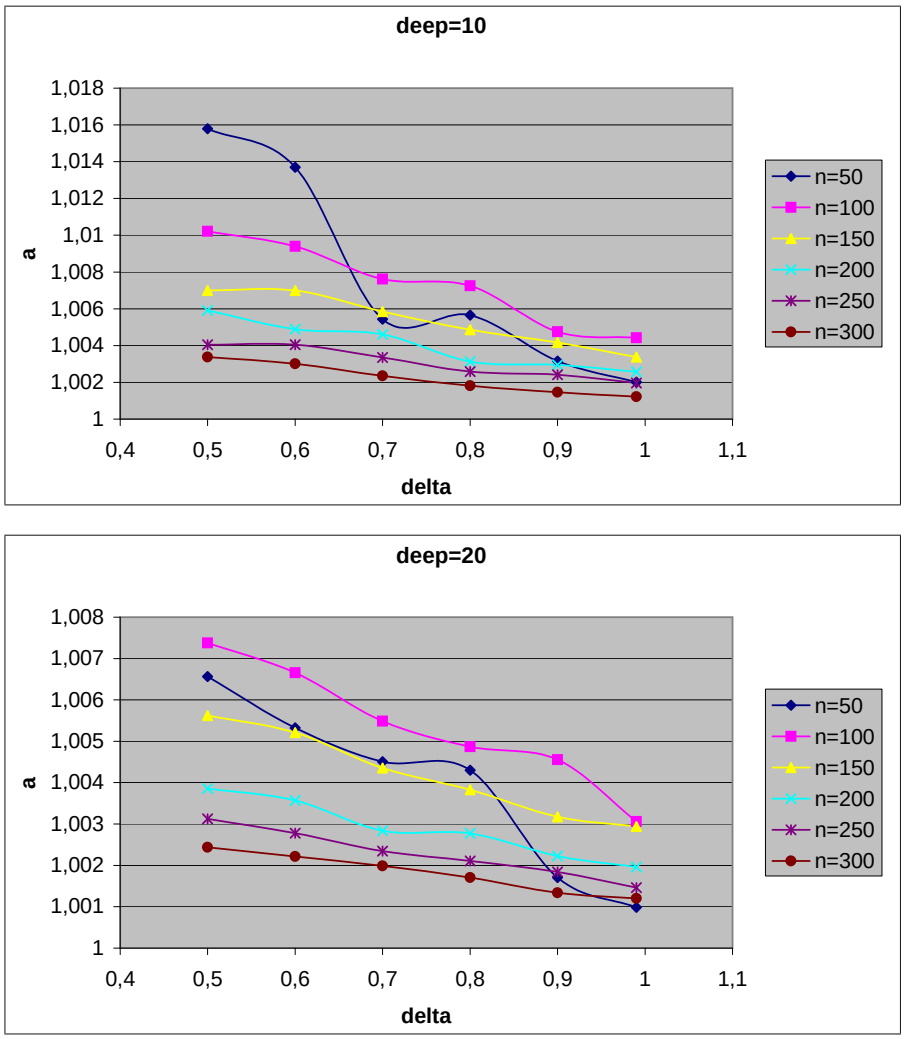


Figure B.11. Approximation factor constants of LLL_XD_DEEP in *delta* (1).

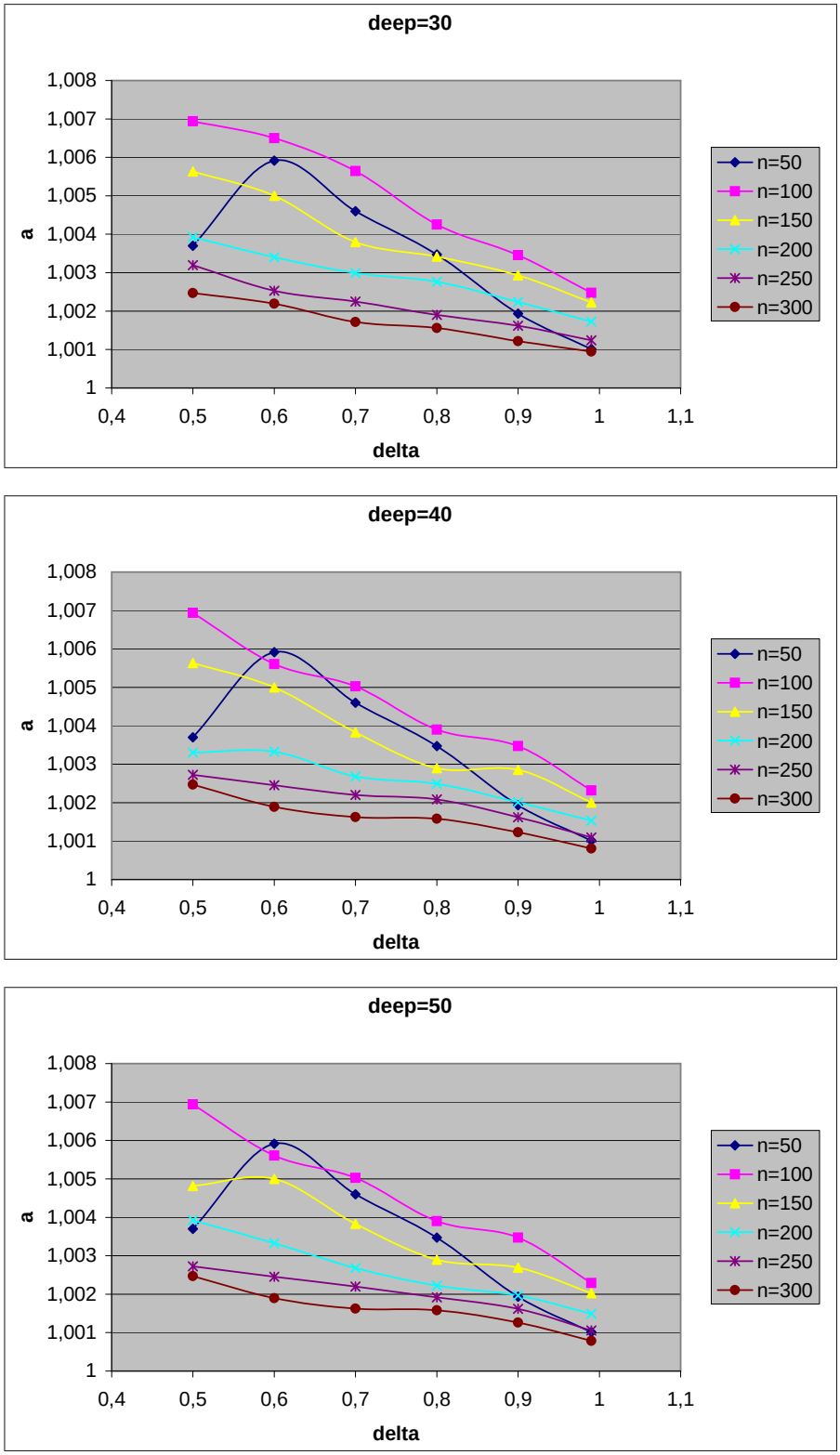


Figure B.12. Approximation factor constants of LLL_XD_DEEP in *delta* (2).

APPENDIX C

BKZ_XD GRAPHS

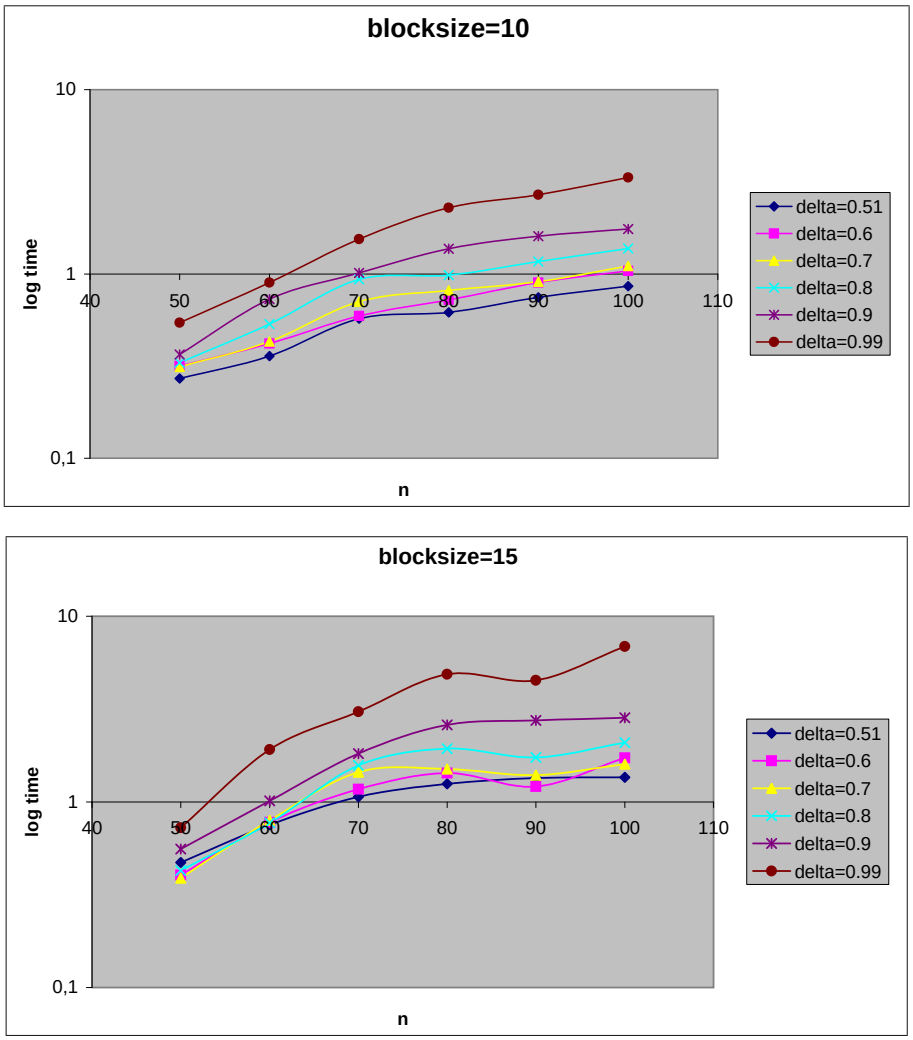


Figure C.1. Running times of BKZ_XD in *n* (1).

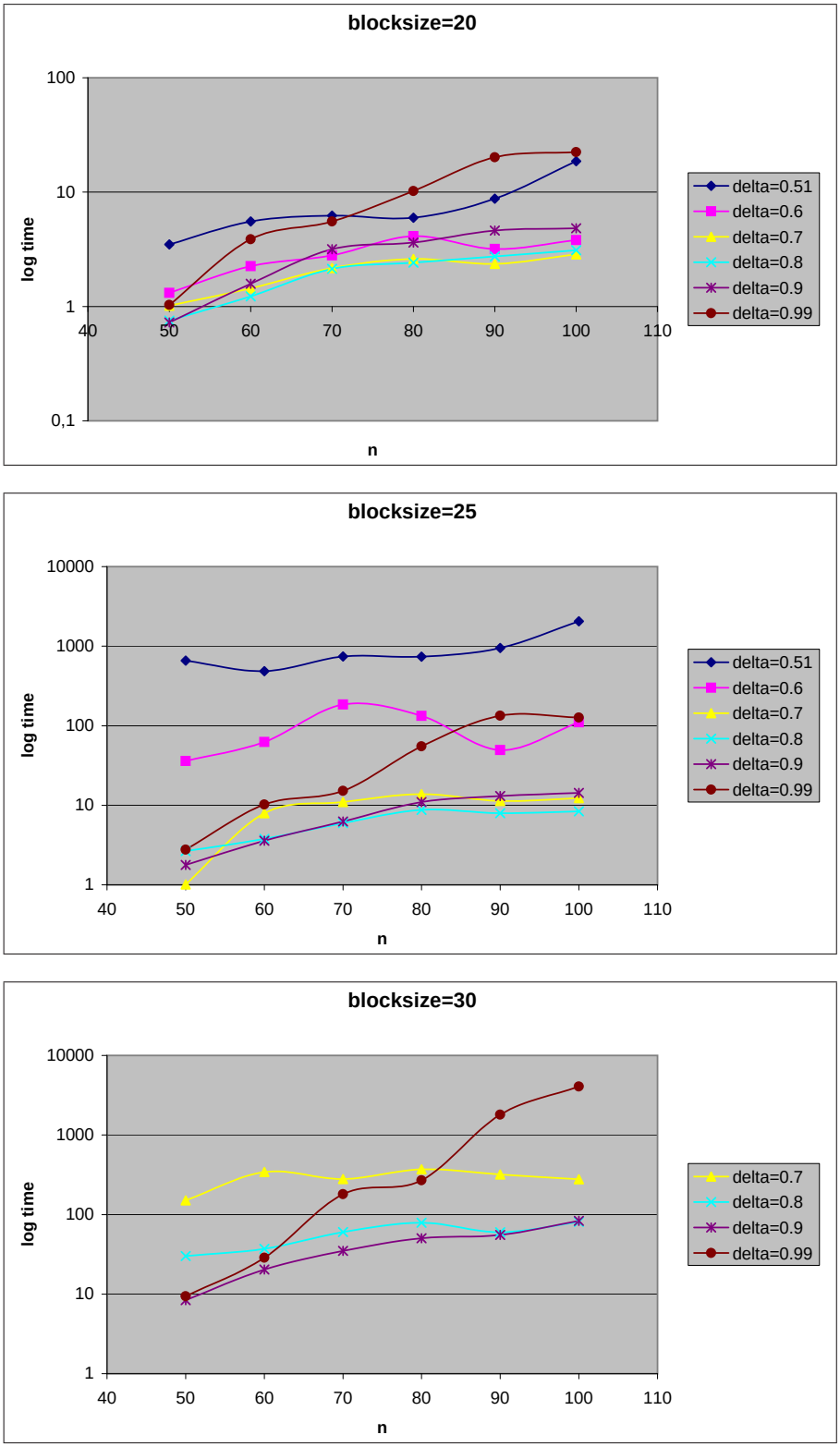


Figure C.2. Running times of BKZ_XD in n (2).

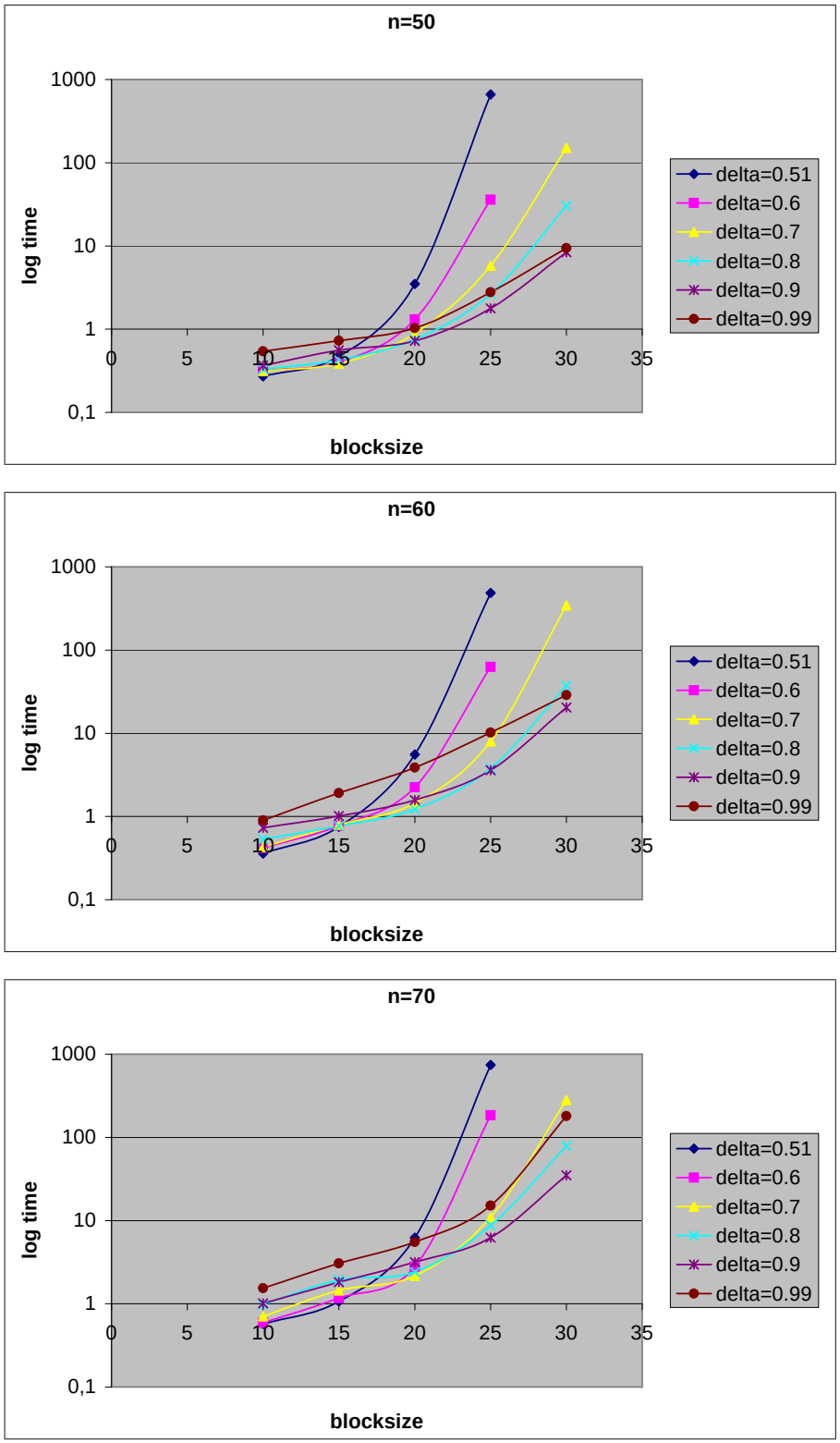


Figure C.3. Running times of BKZ_XD in *blocksize* (1).

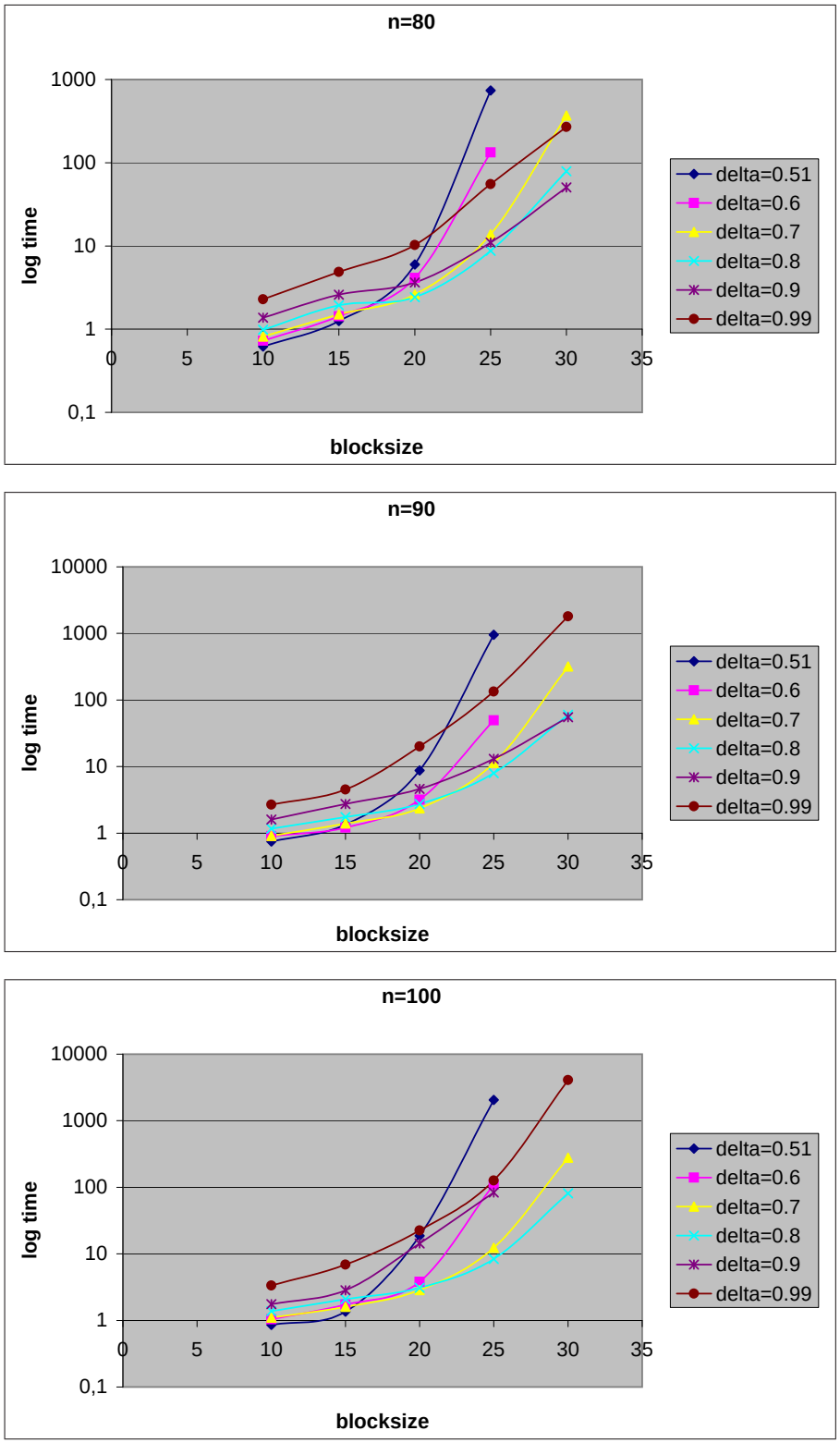


Figure C.4. Running times of BKZ_XD in *blocksize* (2).

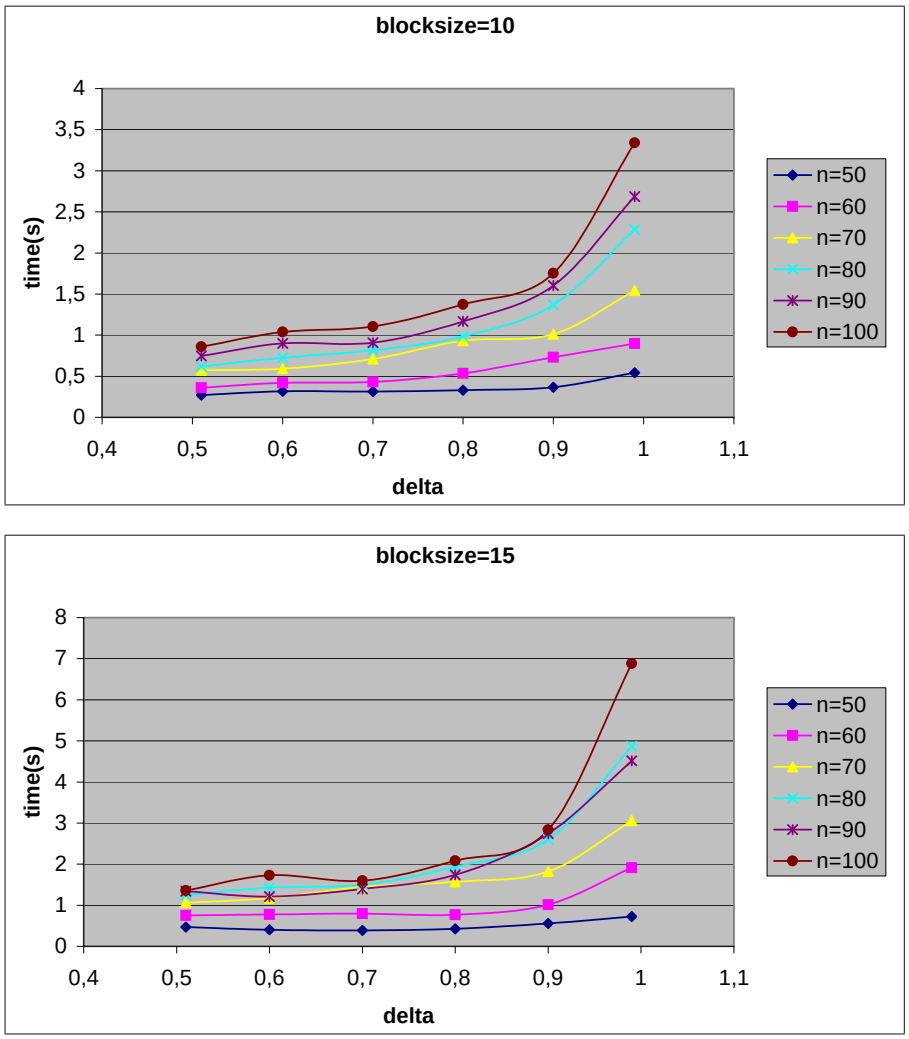


Figure C.5. Running times of BKZ_XD in *delta* (1).

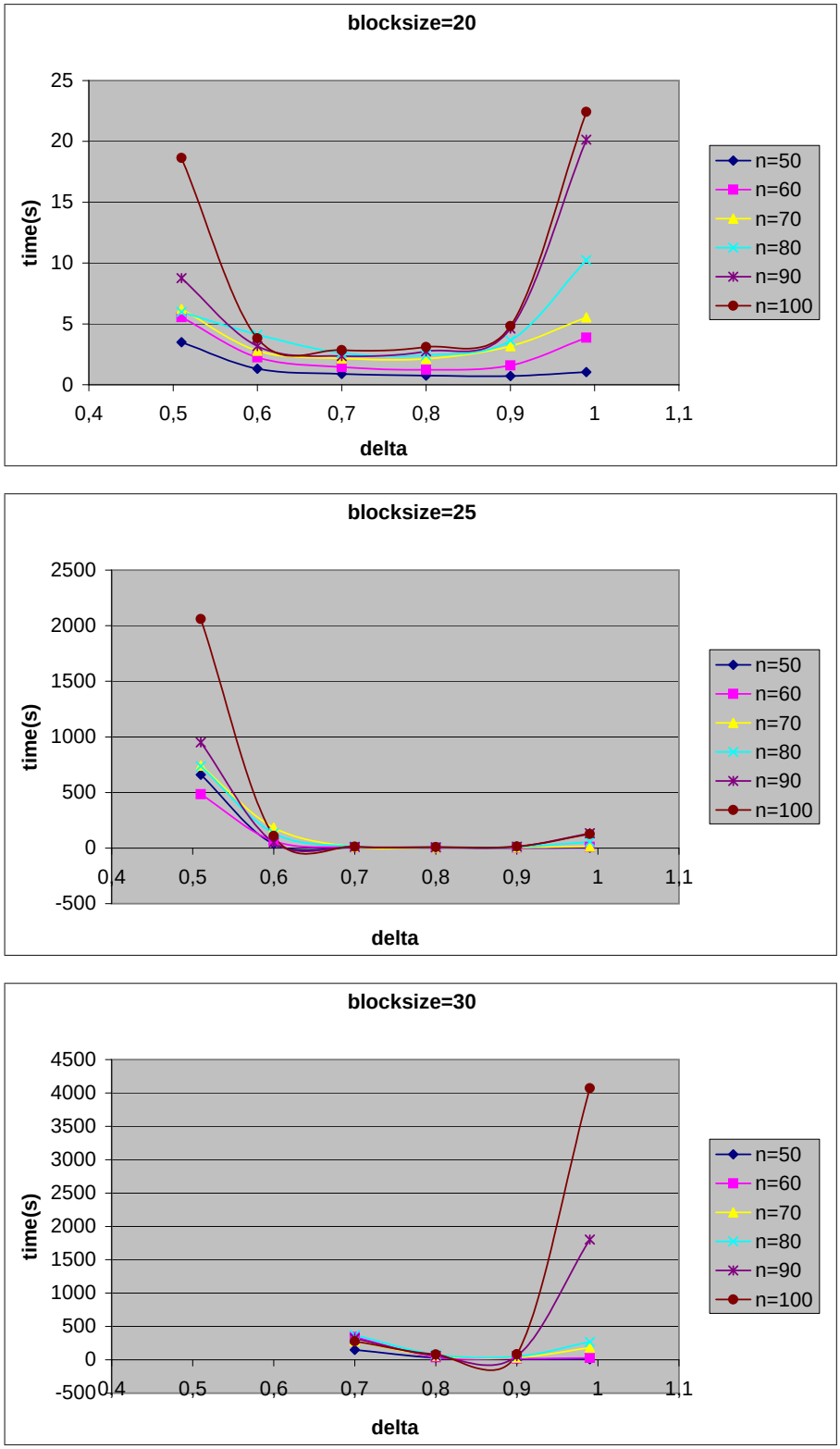


Figure C.6. Running times of BKZ_XD in *delta* (2).

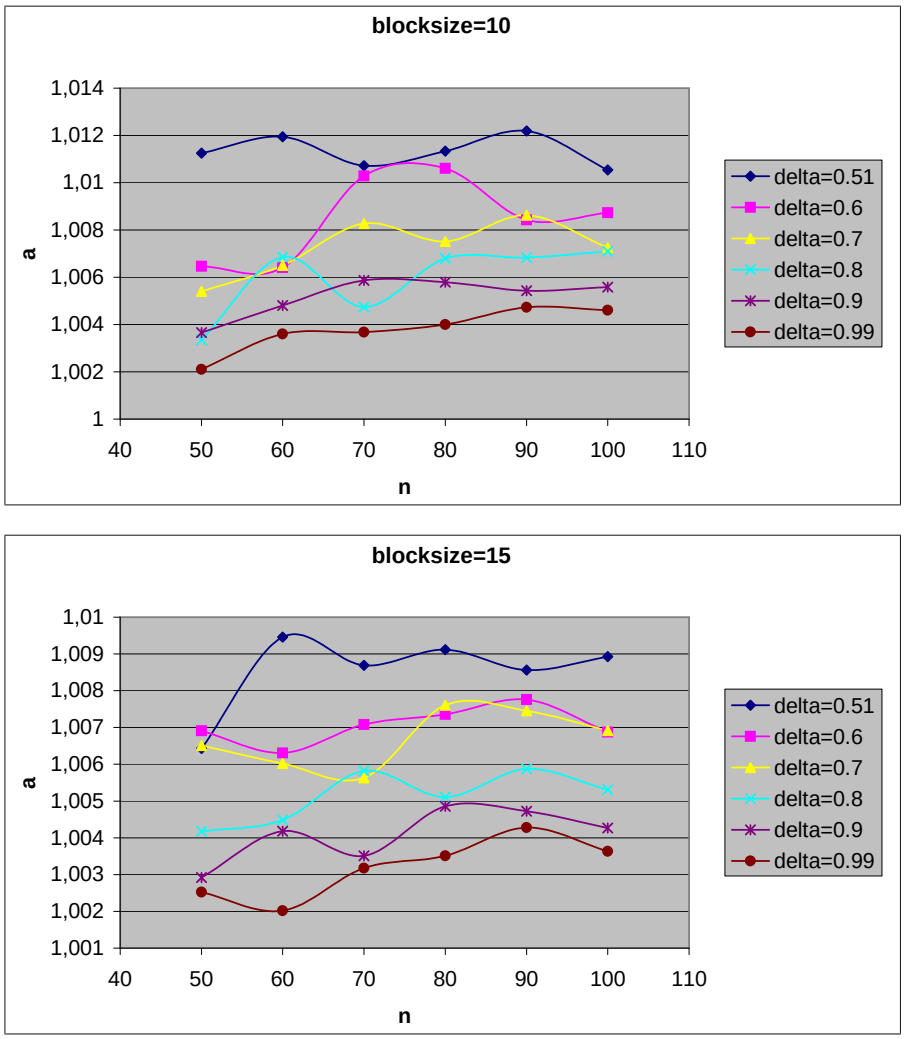


Figure C.7. Approximation factor constants of BKZ_XD in *n* (1).

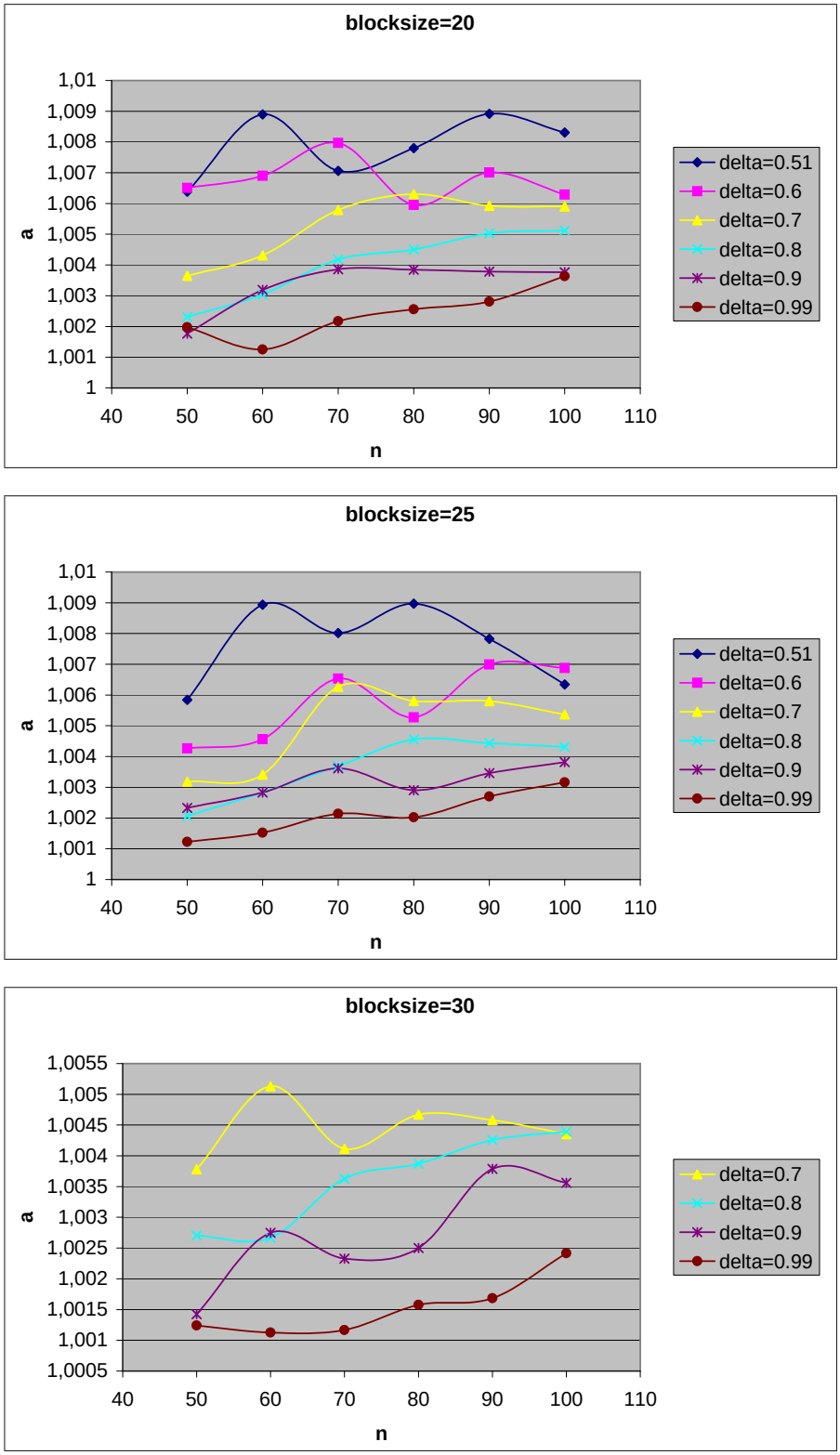


Figure C.8. Approximation factor constants of BKZ_XD in n (2).

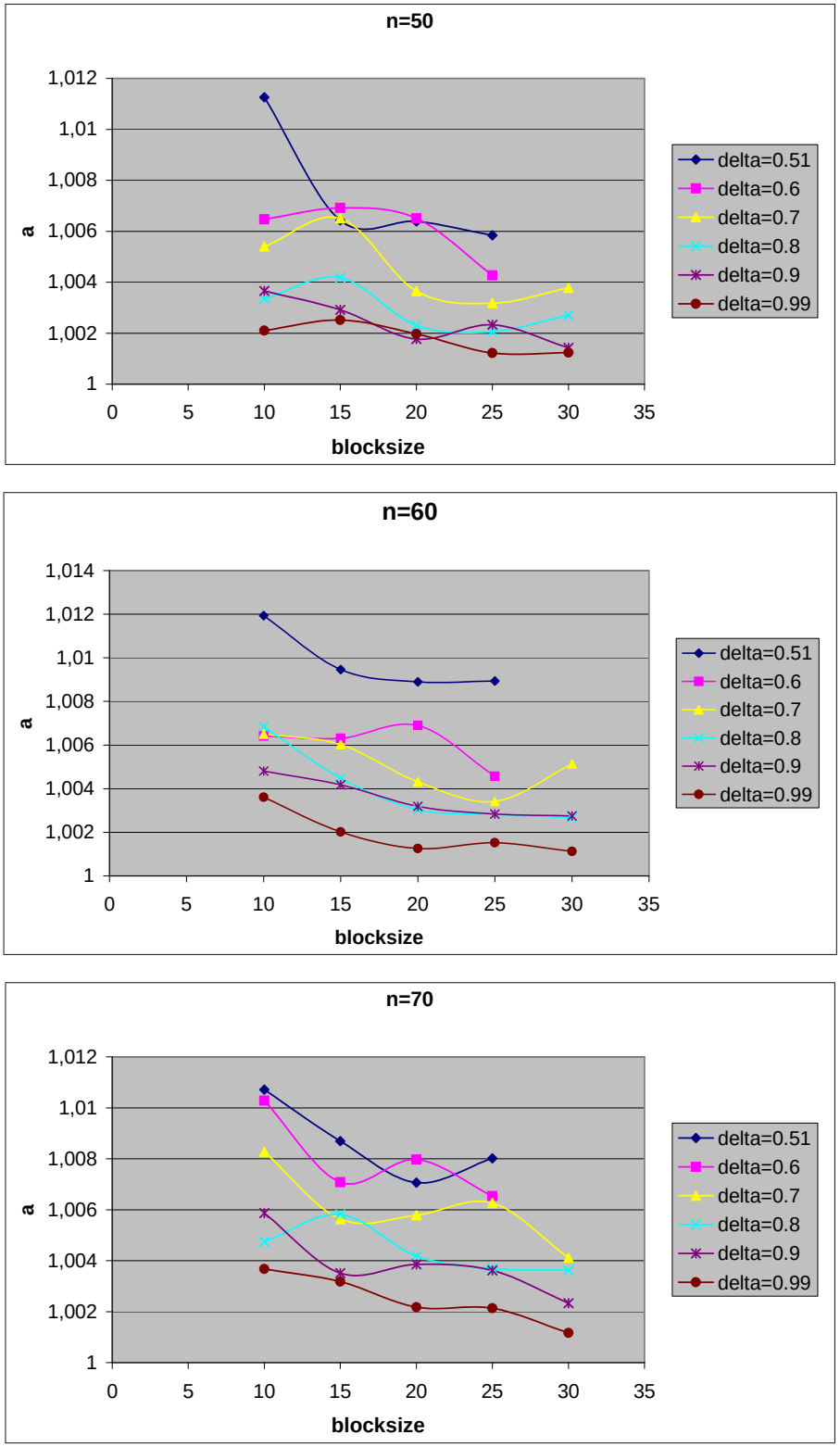


Figure C.9. Approximation factor constants of BKZ_XD in *blocksize* (1).

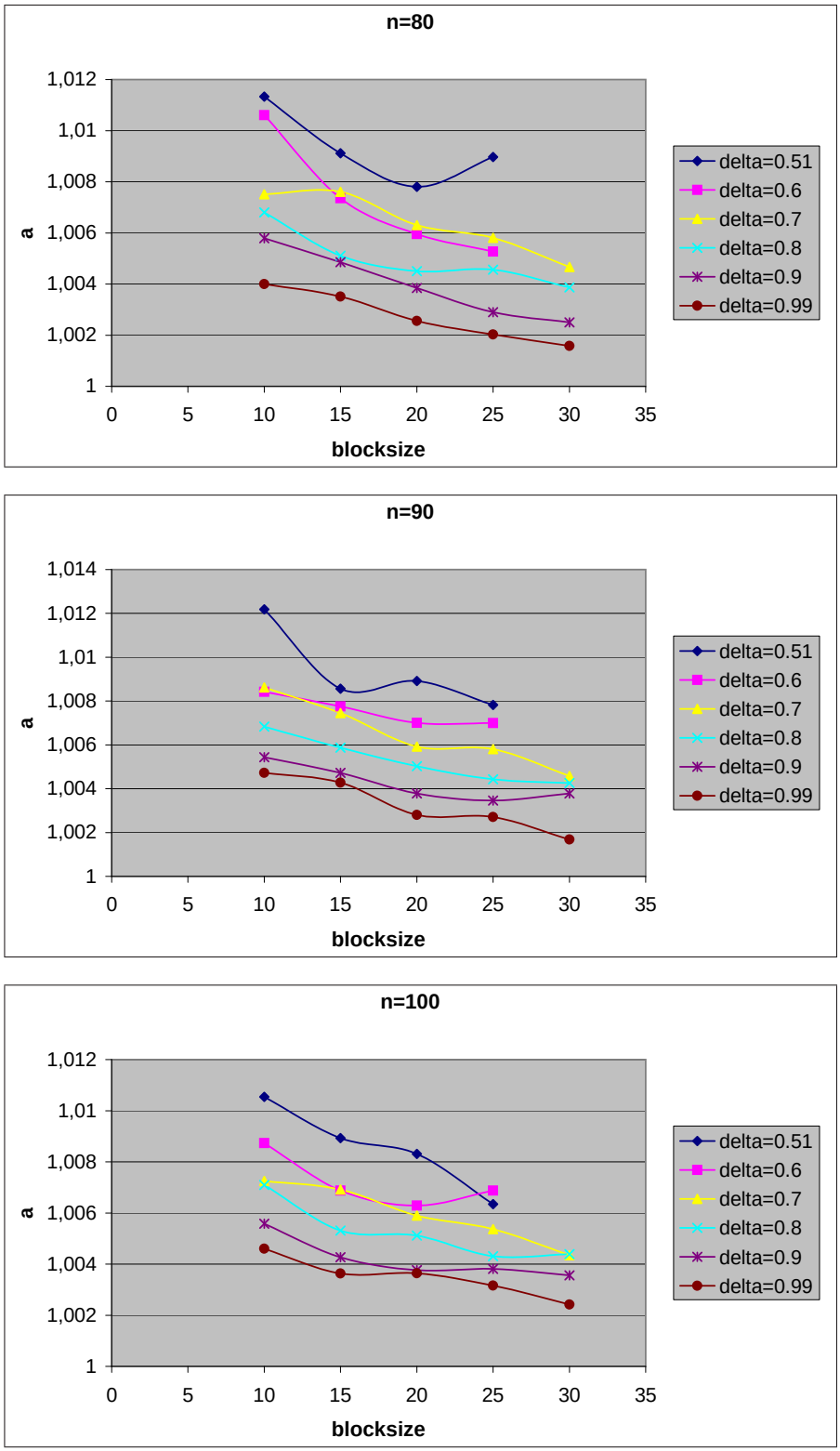


Figure C.10. Approximation factor constants of BKZ_XD in *blocksize* (2).

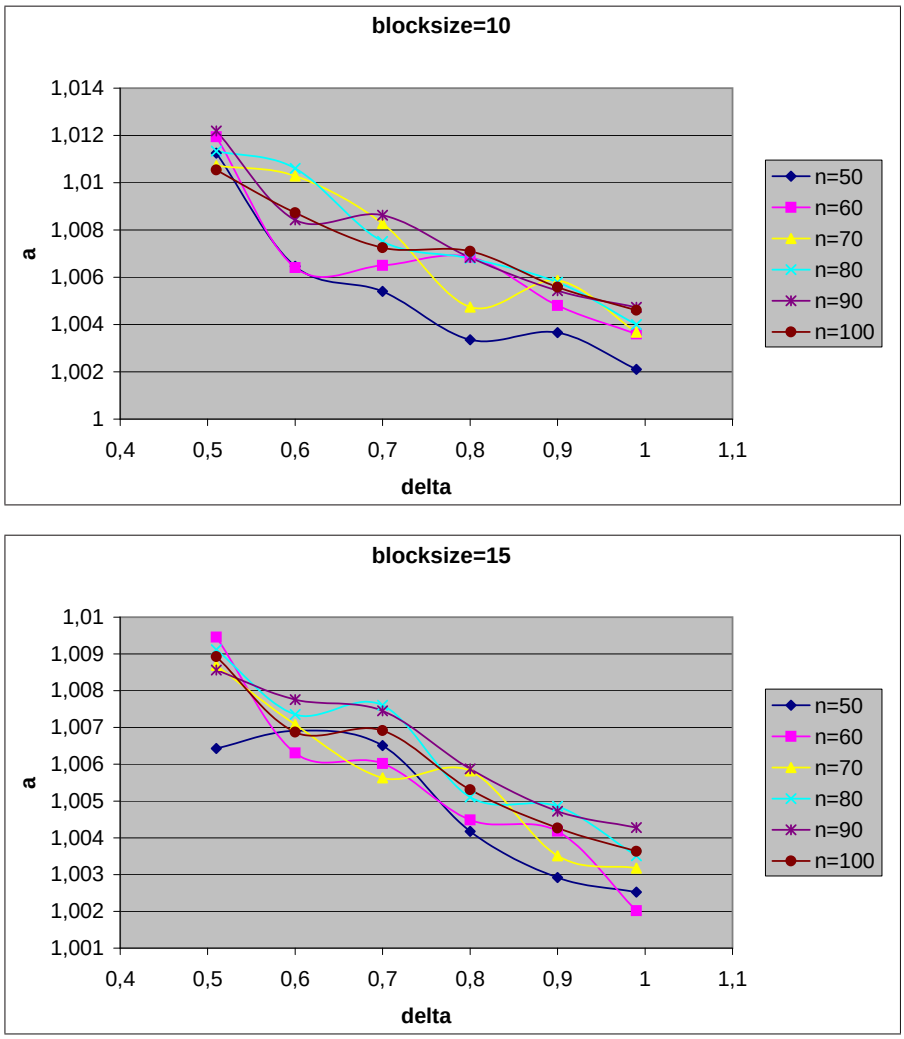


Figure C.11. Approximation factor constants of BKZ_XD in *delta* (1).

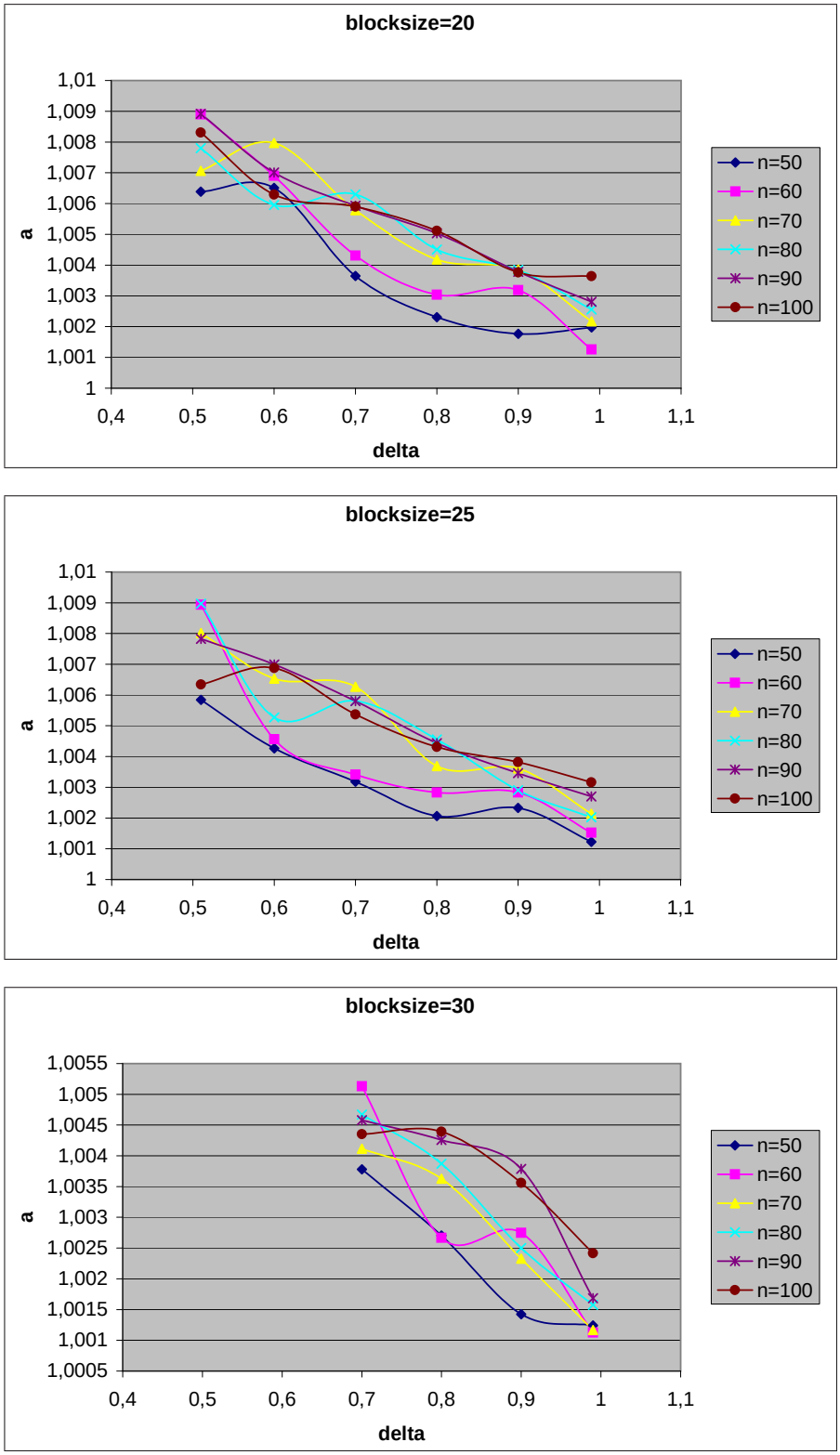
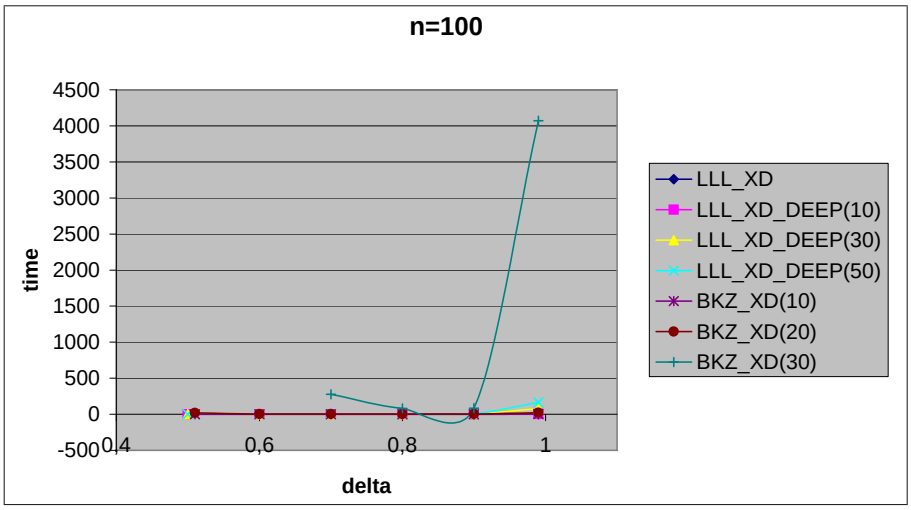


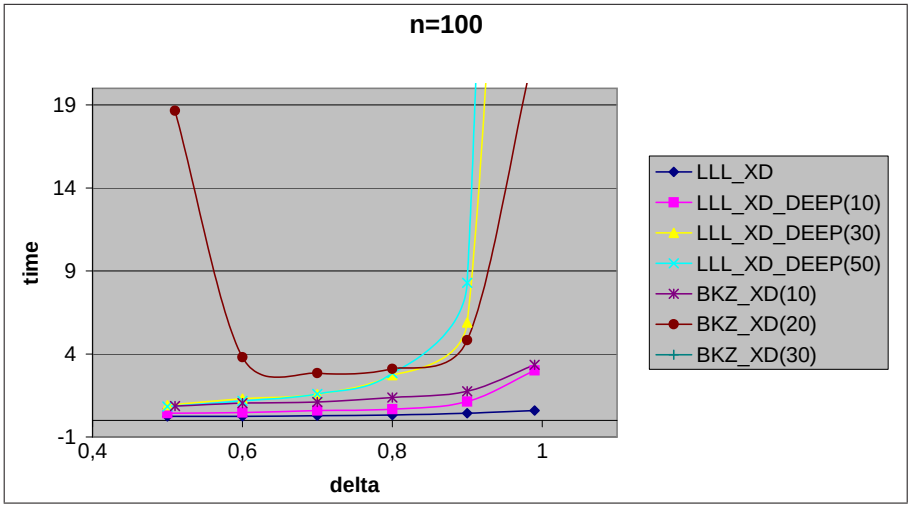
Figure C.12. Approximation factor constants of BKZ_XD in *delta* (2).

APPENDIX D

COMPARISON GRAPHS

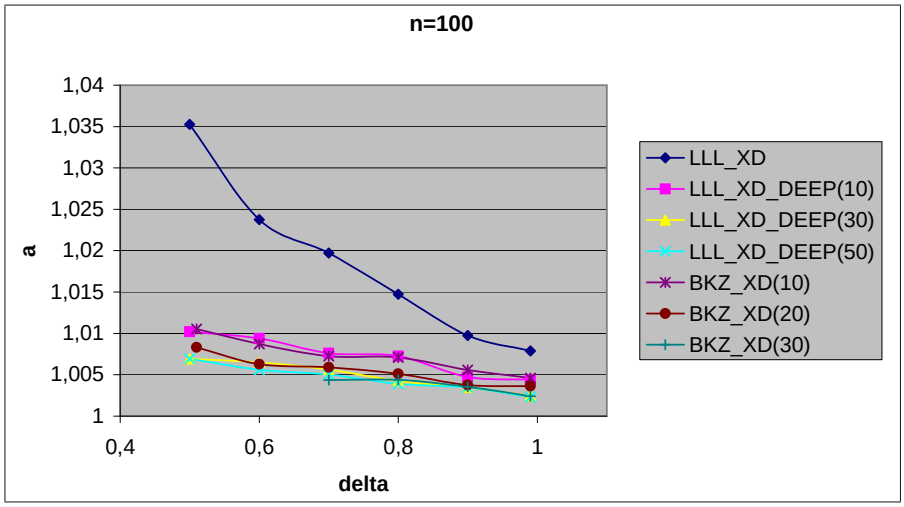


(a) fixed $n = 100$

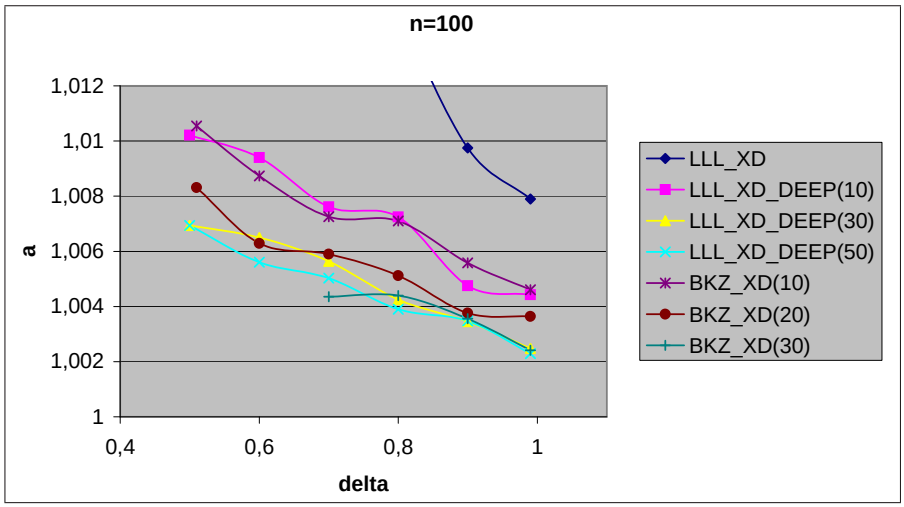


(b) a closer look

Figure D.1. Comparison of running times.



(a) fixed $n = 100$



(b) a closer look

Figure D.2. Comparison of approximation factor constants.