Proceedings of ANTS VII (Jul 23–28, 2006, Berlin, Germany)
F. Hess, S. Pauli and M. Pohst (Ed.), vol. 4076 of Lecture Notes in Computer Science, pages 238–256

CORRECTED VERSION

LLL on the Average

Phong Q. Nguyen*[1] and Damien Stehlé[2]

[1] CNRS & École normale supérieure, DI, 45 rue d'Ulm, 75005 Paris, France.
http://www.di.ens.fr/~pnguyen/
[2] Univ. Nancy 1/LORIA, 615 rue du J. Botanique, 54602 Villers-lès-Nancy, France.
stehle@maths.usyd.edu.au – http://www.loria.fr/~stehle/

Abstract. Despite their popularity, lattice reduction algorithms remain mysterious in many ways. It has been widely reported that they behave much more nicely than what was expected from the worst-case proved bounds, both in terms of the running time and the output quality. In this article, we investigate this puzzling statement by trying to model the average case of lattice reduction algorithms, starting with the celebrated Lenstra-Lenstra-Lovász algorithm (L^3). We discuss what is meant by lattice reduction on the average, and we present extensive experiments on the average case behavior of L^3, in order to give a clearer picture of the differences/similarities between the average and worst cases. Our work is intended to clarify the practical behavior of L^3 and to raise theoretical questions on its average behavior.

1 Introduction

Lattices are discrete subgroups of $\mathbb{R}^n$. A basis of a lattice L is a set of $d \leq n$ linearly independent vectors $\mathbf{b}_1, \ldots, \mathbf{b}_d$ in $\mathbb{R}^n$ such that L is the set $L[\mathbf{b}_1, \ldots, \mathbf{b}_d] = \left\{\sum_{i=1}^{d} x_i \mathbf{b}_i, x_i \in \mathbb{Z}\right\}$ of all integer linear combinations of the $\mathbf{b}_i$'s. The integer d matches the dimension of the linear span of L: it is called the dimension of the lattice L. A lattice has infinitely many bases (except in trivial dimension ≤ 1), but some are more useful than others. The goal of *lattice reduction* is to find interesting lattice bases, such as bases consisting of reasonably short and almost orthogonal vectors. Finding good reduced bases has proved invaluable in many fields of computer science and mathematics (see [9, 14]), particularly in cryptology (see [22, 24]).

The first lattice reduction algorithm in arbitrary dimension is due to Hermite [15]. It was introduced to show the existence of Hermite's constant and of lattice bases with bounded orthogonality defect. Very little is known on the complexity of Hermite's algorithm: the algorithm terminates, but its polynomial-time complexity remains an open question. The subject had a revival with Lenstra's celebrated work on integer programming [19, 20], which used an approximate

* The work described in this article has in part been supported by the Commission of the European Communities through the IST program under contract IST-2002-507932 ECRYPT.

variant of Hermite's algorithm. Lenstra's variant was only polynomial-time for fixed dimension, which was however sufficient in [19]. This inspired Lovász to develop a polynomial-time variant of the algorithm, which reached a final form in [18] where Lenstra, Lenstra and Lovász applied it to factor rational polynomials in polynomial time, from whom the name L^3 comes. Further refinements of L^3 were later proposed, notably by Schnorr [27, 28]. Currently, the most efficient provable variant of L^3 known in case of large entries, called L^2, is due to Nguyen and Stehlé [23], and is based on floating-point arithmetic. Like L^3, it can be viewed as a relaxed version of Hermite's algorithm.

Our Contribution. One of the main reasons why lattice reduction has proved invaluable in many fields is the widely reported experimental fact that lattice reduction algorithms, L^3 in particular, behave much more nicely than what could be expected from the worst-case proved bounds, both in terms of the running time and the output quality. However, to our knowledge, this mysterious phenomenon has never been described in much detail. In this article, we try to give a clearer picture and to give heuristic arguments that explain the situation. We start by discussing what is meant by the average case of lattice reduction, which is related to notions of random lattices and random bases. We then focus on L^3. Regarding the output quality, it seems as if the only difference between the average and worst cases of L^3 in high dimension is a change of constants: while the worst-case behavior of L^3 is closely related to Hermite's constant in dimension two $\gamma_2 = \sqrt{4/3}$, the average case involves a smaller constant whose value is only known experimentally: ≈ 1.04. So while L^3 behaves better than expected, it does not behave that much better: the approximation factors seem to remain exponential in d. Regarding the running time, there is no surprise for the so-called integer version of L^3, except when the input lattice has a special shape such as knapsack-type lattices. However, there can be significant changes with the floating-point variants of L^3. We give a family of bases for which the average running time should be asymptotically close to the worst-case bound, and explain why for reasonable input sizes the executions are faster.

Applications. Guessing the quality of the bases output by L^3 is very important for several reasons. First, all lattice reduction algorithms known rely on L^3 at some stage and their behavior is therefore strongly related to that of L^3. A better understanding of their behavior should provide a better understanding of stronger reduction algorithms such as Schnorr's BKZ [27] and is thus useful to estimate the hardness of lattice problems (which is used in several public-key cryptosystems, such as NTRU [16] and GGH [11]). Besides, if after running L^3, one obtains a basis which is worse than expected, then one should randomize the basis and run L^3 again. Another application comes from the so-called floating-point (*fp* for short) versions of L^3. These are very popular in practice because they are usually much faster. They can however prove tricky to use because they require tuning: if the precision used in *fp*-arithmetic is not chosen carefully, the algorithm may no longer terminate, and if it terminates, it may not give an L^3-reduced basis. On the other hand, the higher the precision, the slower the execution. Choosing the right precision for *fp*-arithmetic is thus important in

practice and it turns out to be closely related to the average-case quality of the bases output by the L^3 algorithm.

The table below sums up our results, for d-dimensional lattices whose initial basis vectors are of lengths smaller than B, with $n = \Theta(d)$ and $d = O(\log B)$.

	$\frac{\|\mathbf{b}_1\|}{(\det L)^{1/d}}$	Running time of L^2	Required prec. for L^2
Worst-case bound	$(4/3)^{d/4}$	$O(d^5 \log^2 B)$	$\approx 1.58d + o(d)$
Average-case estim.	$(1.02)^d$	$O(d^4 \log^2 B) \rightarrow O(d^5 \log^2 B)$	$0.25d + o(d)$

ROAD MAP. In Section 2 we provide necessary background on L^3. We discuss random lattices and random bases in Section 3. Then we describe our experimental observations on the quality of the computed bases (Section 4), the running time (Section 5) and the numerical behavior (Section 6).

ADDITIONAL MATERIAL. All experiments were performed with `fplll-1.2`, available at `http://www.loria.fr/~stehle/practLLL.html`. The data used to draw the figures of the paper and some others are also available at this URL.

2 Background

NOTATION. All logarithms are in base 2. Let $\|\cdot\|$ and $\langle\cdot,\cdot\rangle$ be the Euclidean norm and inner product of $\mathbb{R}^n$. The notation $\lceil x \rfloor$ denotes a closest integer to x. Bold variables are vectors. All the lattices we consider are integer lattices, as usual. All our complexity results are given for the bit complexity model, without fast integer arithmetic. Our *fpa*-model is a smooth extension of the IEEE-754 standard, as provided by NTL [30] (RR class) and MPFR [26].

We recall basic notions from algorithmic geometry of numbers (see [22]).

First minimum. If L is a lattice, we denote by $\lambda(L)$ its *first minimum*.

Gram matrix. Let $\mathbf{b}_1, \ldots, \mathbf{b}_d$ be vectors. Their *Gram matrix* $G(\mathbf{b}_1, \ldots, \mathbf{b}_d)$ is the $d \times d$ symmetric matrix $(\langle \mathbf{b}_i, \mathbf{b}_j \rangle)_{1 \leq i,j \leq d}$ formed by all the inner products.

Gram-Schmidt orthogonalization. Let $\mathbf{b}_1, \ldots, \mathbf{b}_d$ be linearly independent vectors. The *Gram-Schmidt orthogonalization* (GSO) $[\mathbf{b}_1^*, \ldots, \mathbf{b}_d^*]$ is the orthogonal family defined as follows: $\mathbf{b}_i^*$ is the component of $\mathbf{b}_i$ orthogonal to the linear span of $\mathbf{b}_1, \ldots, \mathbf{b}_{i-1}$. We have $\mathbf{b}_i^* = \mathbf{b}_i - \sum_{j=1}^{i-1} \mu_{i,j} \mathbf{b}_j^*$ where $\mu_{i,j} = \langle \mathbf{b}_i, \mathbf{b}_j^* \rangle / \|\mathbf{b}_j^*\|^2$. For $i \leq d$, we let $\mu_{i,i} = 1$. The lattice L spanned by the $\mathbf{b}_i$'s satisfies $\det L = \prod_{i=1}^{d} \|\mathbf{b}_i^*\|$. The GSO family depends on the order of the vectors. If the $\mathbf{b}_i$'s are integer vectors, the $\mathbf{b}_i^*$'s and the $\mu_{i,j}$'s are rational. In what follows, the *GSO family* denotes the $\mu_{i,j}$'s, together with the quantities $r_{i,j}$'s defined as: $r_{i,i} = \|\mathbf{b}_i^*\|^2$ and $r_{i,j} = \mu_{i,j} r_{j,j}$ for $j < i$.

Size-reduction. A basis $[\mathbf{b}_1, \ldots, \mathbf{b}_d]$ is *size-reduced* with factor $\eta \geq 1/2$ if its GSO family satisfies $|\mu_{i,j}| \leq \eta$ for all $j < i$. The i-th vector $\mathbf{b}_i$ is *size-reduced* if $|\mu_{i,j}| \leq \eta$ for all $j < i$. Size-reduction usually refers to $\eta = 1/2$, but it is essential for *fp* variants of L^3 to allow larger η.

L^3-reduction. A basis $[\mathbf{b}_1, \ldots, \mathbf{b}_d]$ is L^3-reduced with factor (δ, η) with $1/4 < \delta \leq 1$ and $1/2 \leq \eta < \sqrt{\delta}$ if the basis is η-size-reduced and if its GSO satisfies the $(d-1)$ Lovász conditions $(\delta - \mu_{\kappa,\kappa-1}^2) r_{\kappa-1,\kappa-1} \leq r_{\kappa,\kappa}$ (or equivalently $\delta \|\mathbf{b}_{\kappa-1}^*\|^2 \leq \|\mathbf{b}_\kappa^* + \mu_{\kappa,\kappa-1} \mathbf{b}_{\kappa-1}^*\|^2$), which implies that the $\|\mathbf{b}_\kappa^*\|$'s never

drop too much. Such bases have useful properties (see [18]), like providing approximations to the shortest and closest vector problems. In particular, the first vector is relatively short: $\|\mathbf{b}_1\| \leq \beta^{(d-1)/4}(\det L)^{1/d}$, where $\beta = 1/(\delta - \eta^2)$. And the first basis vector is at most exponentially far away from the first minimum: $\|\mathbf{b}_1\| \leq \beta^{(d-1)/2}\lambda(L)$. L^3-reduction usually refers to the factor $(3/4, 1/2)$ initially chosen in [18], in which case $\beta = 2$. But the closer (δ, η) is to $(1, 1/2)$, the shorter $\mathbf{b}_1$ should be. In practice, one usually selects $\delta \approx 1$ and $\eta \approx 1/2$, so that $\beta \approx 4/3$ and therefore $\|\mathbf{b}_1\| \lesssim (4/3)^{(d-1)/4}(\det L)^{1/d}$. The L^3 algorithm obtains in polynomial time a $(\delta, 1/2)$-L^3-reduced basis where $\delta < 1$ can be chosen arbitrarily close to 1. The L^2 algorithm achieves a factor (δ, η), where $\delta < 1$ can be arbitrarily close to 1 and $\eta > 1/2$ arbitrarily close to $1/2$.

Input: A basis $[\mathbf{b}_1, \ldots, \mathbf{b}_d]$ and $\delta \in (1/4, 1)$.
Output: An L^3-reduced basis with factor $(\delta, 1/2)$.
1. Compute the rational GSO, i.e., all the $\mu_{i,j}$'s and $r_{i,i}$'s.
2. κ:=2. While $\kappa \leq d$ do
3. Size-reduce $\mathbf{b}_\kappa$ using the algorithm of Figure 2, that updates the GSO.
4. κ':=κ. While $\kappa \geq 2$ and $\delta r_{\kappa-1,\kappa-1} \geq r_{\kappa',\kappa'} + \sum_{i=\kappa-1}^{\kappa'-1} \mu_{\kappa',i}^2 r_{i,i}$, do κ:=$\kappa - 1$.
5. For $i = 1$ to $\kappa - 1$, $\mu_{\kappa,i}$:=$\mu_{\kappa',i}$. Insert $\mathbf{b}_{\kappa'}$ right before $\mathbf{b}_\kappa$.
6. κ:=$\kappa + 1$.
7. Output $[\mathbf{b}_1, \ldots, \mathbf{b}_d]$.

Fig. 1. The L^3 Algorithm.

Input: A basis $[\mathbf{b}_1, \ldots, \mathbf{b}_d]$, its GSO and an index κ.
Output: The basis with $\mathbf{b}_\kappa$ size-reduced and the updated GSO.
1. For $i = \kappa - 1$ down to 1 do
2. $\mathbf{b}_\kappa$:=$\mathbf{b}_\kappa - \lceil \mu_{\kappa,i} \rfloor \mathbf{b}_i$.
3. For $j = 1$ to i do $\mu_{\kappa,j}$:=$\mu_{\kappa,j} - \lceil \mu_{\kappa,i} \rfloor \mu_{i,j}$.
4. Update the GSO accordingly.

Fig. 2. The size-reduction algorithm.

The L^3 algorithm. The L^3 algorithm [18] is described in Figure 1. It computes an L^3-reduced basis in an iterative fashion: the index κ is such that at any stage of the algorithm, the truncated basis $[\mathbf{b}_1, \ldots, \mathbf{b}_{\kappa-1}]$ is L^3-reduced. At each loop iteration, κ is either incremented or decremented: the loop stops when κ reaches the value $d + 1$, in which case the entire basis $[\mathbf{b}_1, \ldots, \mathbf{b}_d]$ is L^3-reduced.

If L^3 terminates, it is clear that the output basis is L^3-reduced. What is less clear *a priori* is why L^3 has a polynomial-time complexity. A standard argument shows that each swap decreases the quantity $\Delta = \prod_{i=1}^{d} \|\mathbf{b}_i^*\|^{2(d-i+1)}$ by at least a factor $\delta < 1$, while $\Delta \geq 1$ because the $\mathbf{b}_i$'s are integer vectors and Δ can be viewed as a product of squared volumes of lattices spanned by some of the $\mathbf{b}_i$'s. This proves that there are $O(d^2 \log B)$ swaps, and therefore loop iterations, where B is an upper bound on the norms of the input basis vectors. It remains to estimate the cost of each loop iteration. This cost turns out to be dominated by $O(dn)$ arithmetic operations on the basis matrix and GSO coefficients $\mu_{i,j}$ and $r_{i,i}$ which are rational numbers of bit-length $O(d \log B)$. Thus, the overall complexity of L^3 is $O((d^2 \log B) \cdot dn \cdot (d \log B)^2)) = O(d^5 n \log^3 B)$.

L^3 with *fpa*. The cost of L^3 is dominated by the operations on the GSO coefficients which are rationals with huge numerators and denominators. It is therefore tempting to replace the exact GSO coefficients by *fp* approximations. But doing so in a straightforward manner leads to numerical anomalies. The algorithm is no longer guaranteed to be polynomial-time: it may not even terminate. And if ever it terminates, the output basis may not be L^3-reduced. The main number theory computer packages [7, 21, 30] contain heuristic *fp*-variants of L^3 *à la* Schnorr-Euchner [29] suffering from stability problems. On the theoretic side, the fastest provable *fp* variant of L^3 is Nguyen-Stehlé's L^2 [23], whose running time is $O(d^4 n(d + \log B) \log B)$. The main algorithmic differences with Schnorr-Euchner's *fp* L^3 are that the integer Gram matrix is updated during the execution (thus avoiding cancellations while computing scalar products with *fpa*), and that the size-reduction algorithm is replaced by a lazy variant (this idea was already in Victor Shoup's NTL code). In L^2, the worst-case required precision for *fpa* is $\leq 1.59d + o(d)$. The proved variant of `fplll-1.2` implements L^2.

3 Random Lattices

In this section, we give the main methods known to generate random lattices and random bases, and describe the random bases we use in our experiments.

3.1 Random Lattices

When experimenting with L^3, it seems natural to work with random lattices, but what is a random lattice? From a practical point of view, one could just select randomly generated lattices of interest, such as lattices used in cryptography or in algorithmic number theory. This would already be useful but one might argue that it would be insufficient to draw conclusions, because such lattices may not be considered random in a mathematical sense. For instance, in many cryptanalyses, one applies reduction algorithms to lattices whose first minimum is much shorter than all the other minima.

From a mathematical point of view, there is a natural notion of random lattice, which follows from a measure on n-dimensional lattices with determinant 1 introduced by Siegel [31] back in 1945, to provide an alternative proof of the Minkowski-Hlwaka theorem. Let $X_n = SL_n(\mathbb{R})/SL_n(\mathbb{Z})$ be the space of (full-rank) lattices in $\mathbb{R}^n$ modulo scale. The group $G = SL_n(\mathbb{R})$ possesses a unique (up to scale) bi-invariant Haar measure, which can be thought of as the measure it inherits as a hypersurface in $\mathbb{R}^{n^2}$. When mapping G to the quotient $X_n = G/SL_n(\mathbb{Z})$, the Haar measure projects to a finite measure μ on the space X_n which we can normalize to have total volume 1. This measure μ is G-invariant: if $A \subseteq X_n$ is measurable and $g \in G$, then $\mu(A) = \mu(gA)$. In fact, μ can be characterized as the unique G invariant Borel probability measure on X_n. This gives rise to a natural notion of random lattices. The recent articles [2, 4, 12] propose efficient ways to generate lattices which are random in this sense. For instance, Goldstein and Mayer [12] show that for large N, the

(finite) set $\mathcal{L}_{n,N}$ of n-dimensional integer lattices of determinant N is uniformly distributed in X_n in the following sense: given any measurable subset $A \subseteq X_n$ whose boundary has zero measure with respect to μ, the fraction of lattices in $\mathcal{L}_{n,N}/N^{1/n}$ that lie in A tends to $\mu(A)$ as N tends to infinity.

Thus, to generate lattices that are random in a natural sense, it suffices to generate uniformly at random a lattice in $\mathcal{L}_{n,N}$ for large N. This is particularly easy when N is prime. Indeed, when p is a large prime, the vast majority of lattices in $\mathcal{L}_{n,p}$ are lattices spanned by row matrices of the following form:

$$R_p^n = \begin{pmatrix} p & 0 & 0 & \dots & 0 \\ x_1 & 1 & 0 & \dots & 0 \\ x_2 & 0 & 1 & \ddots & \vdots \\ \vdots & \vdots & \ddots & \ddots & 0 \\ x_{n-1} & 0 & \dots & 0 & 1 \end{pmatrix},$$

where the x_i's are chosen independently and uniformly in $\{0, \dots, p-1\}$.

3.2 Random Bases

Once a lattice has been selected, it would be useful to select a random basis, among the infinitely many bases. This time however, there is no clear definition of what is a random basis, since there is no finite measure on $SL_n(\mathbb{Z})$. Since we mostly deal with integer lattices, one could consider the Hermite normal form (HNF) of the lattice, and argue that this is the basis which gives the least information on the lattice, because it can be computed in polynomial time from any basis. However, it could also be argued that the HNF may have special properties, depending on the lattice. For instance, the HNF of NTRU lattices [16] is already reduced in some sense, and does not look like a random basis at all. A random basis should consist of long vectors: the orthogonality defect should not be bounded, since the number of bases with bounded orthogonality defect is bounded. In other words, a random basis should not be reduced at all.

A heuristic approach was used for the GGH cryptosystem [11]. Namely, a secret basis was transformed into a large public basis of the same lattice by multiplying generators of $SL_n(\mathbb{Z})$ in a random manner. However, it is difficult to control the size of the entries, and it looks hard to obtain theoretical results.

One can devise a less heuristic method as follows. Consider a full-rank integer lattice $L \subseteq \mathbb{Z}^n$. If B is much bigger than $(\det L)^{1/n}$, it is possible to sample efficiently and uniformly points in $L \cap [-B, B]^n$ (see [1]). For instance, if $B = (\det L)/2$, one can simply take an integer linear combination $x_1\boldsymbol{b}_1 + \cdots + x_n\boldsymbol{b}_n$ of a basis, with large coefficients x_i, and reduce the coordinates modulo $\det L = [\mathbb{Z}^n : L]$. That is easy for lattices in the previous set $\mathcal{L}_{n,p}$ where p is prime. Once we have such a sampling procedure, we note that n vectors randomly chosen in such a way will with overwhelming probability be linearly independent. Though they are unlikely to form a lattice basis (rather, they will span a sublattice), one can easily lift such a full-rank set of linearly independent vectors of norm $\leq B$ to a basis made of vectors of norm $\leq B\sqrt{n}/2$ using Babai's nearest plane algorithm [5] (see [1] or [22, Lemma 7.1]). In particular, if one considers the lattices of the

class $\mathcal{L}_{n,p}$, it is easy to generate plenty of bases in a random manner in such a way that all the coefficients of the basis vectors are $\leq p\sqrt{n}/2$.

3.3 Random L^3-Reduced Bases

There are two natural notions of random L^3 bases. One is derived from the mathematical definition. An L^3-reduced basis is necessarily Siegel-reduced (following the definition of [8]), that is, its $\mu_{i,j}$'s and $\|\boldsymbol{b}_i^*\|/\|\boldsymbol{b}_{i+1}^*\|$'s are bounded. This implies [8] that the number of L^3-reduced bases of a given lattice is finite (for any reduction parameters), and can be bounded independently of the lattice. Thus, one could define a random L^3 basis as follows: select a random lattice, and among all the finitely many L^3-reduced bases of that lattice, select one uniformly at random. Unfortunately, the latter process is impractical, but it might be interesting to prove probabilistic statements on such bases. Instead, one could try the following in practice: select a random lattice, then select a random basis, and eventually apply the L^3 algorithm. The output basis will not necessarily be random in the first sense, since the L^3 algorithm may bias the distribution. However, intuitively, it could also be viewed as some kind of random L^3 basis. In the previous process, it is crucial to select a random-looking basis (unlike the HNF of NTRU lattices). For instance, if we run the L^3 algorithm on already reduced (or almost reduced) bases, the output basis will differ from a typical L^3-reduced basis.

3.4 Random Bases in Our Experiments

In our experiments, besides the Goldstein-Mayer [12] bases of random lattices, we considered two other types of random bases. The Ajtai-type bases of dimension d and factor α are given by the rows of a lower triangular random matrix B with:

$$B_{i,i} = \lfloor 2^{(2d-i+1)^\alpha} \rfloor \quad \text{and} \quad B_{j,i} = \text{rand}(-B_{i,i}/2, B_{i,i}/2) \quad \text{for all } j > i.$$

Similar bases have been used by Ajtai in [3] to show the tightness of worst-case bounds of [27]. The bases used in Coppersmith's root-finding method [10] bear some similarities with what we call Ajtai-type bases.

We define the knapsack-type bases as the rows of the $d \times (d+1)$ matrices:

$$\begin{pmatrix} A_1 & 1 & 0 & \dots & 0 \\ A_2 & 0 & 1 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ A_d & 0 & 0 & \dots & 1 \end{pmatrix},$$

where the A_i's are sampled independently and uniformly in $[-B, B]$, for some given bound B. Such bases often occur in practice, e.g., in cryptanalyses of knapsack-based cryptosystems, reconstructions of minimal polynomials and detections of integer relations between real numbers. The behavior of L^3 on this type of bases and on the above R_p^{d+1}'s look alike.

Interestingly, we did not notice any significant change in the output quality or in the geometry of reduced bases between all three types of random bases.

4 The Output Quality of L³

For fixed parameters δ and η, the L^3 and L^2 algorithms output bases $\mathbf{b}_1, \ldots, \mathbf{b}_d$ such that $\|\mathbf{b}^*_{i+1}\|^2/\|\mathbf{b}^*_i\|^2 \geq \beta = 1/(\delta - \eta^2)$ for all $i < d$, which implies that:

$$\|\mathbf{b}_1\| \leq \beta^{(d-1)/4}(\det L)^{1/d} \quad \text{and} \quad \|\mathbf{b}_1\| \leq \beta^{(d-1)/2}\lambda(L).$$

It is easy to prove that these bounds are tight in the worst case: both are reached for some reduced bases of some particular lattices. However, there is a common belief that they are not tight in practice. For instance, Odlyzko wrote in [25]:

This algorithm [...] *usually finds a reduced basis in which the first vector is much shorter than guaranteed* [theoretically]. *(In low dimensions, it has been observed empirically that it usually finds the shortest non-zero vector in a lattice.)*

We argue that the quantity $\|\mathbf{b}_1\|/(\det L)^{1/d}$ remains exponential on the average, but is indeed far smaller than the worst-case bound: for δ close to 1 and η close to 1/2, one should replace $\beta^{1/4} \approx (4/3)^{1/4}$ by ≈ 1.02, so that the approximation factor $\beta^{(d-1)/4}$ becomes $\approx 1.02^d$. As opposed to the worst-case bounds, the ratio $\|\mathbf{b}_1\|/\lambda(L)$ should also be $\approx 1.02^d$ on the average, rather than being the square of $\|\mathbf{b}_1\|/(\det L)^{1/d}$. Indeed, if the Gaussian heuristic holds for a lattice L, then $\lambda(L) \approx \sqrt{\frac{d}{2\pi e}}(\det L)^{1/d}$. The Gaussian heuristic is only a heuristic in general, but it can be proved for random lattices (see [2, 4]), and it is unlikely to be wrong by an exponential factor, unless the lattice is very special.

Heuristic 1 *Let δ be close to 1 and η be close to 1/2. Given as input a random basis of almost any lattice L of sufficiently high dimension (e.g., larger than 40), L^3 and L^2 with parameters δ and η output a basis whose first vector $\mathbf{b}_1$ satisfies $\|\mathbf{b}_1\|/(\det L)^{1/d} \approx (1.02)^d$.*

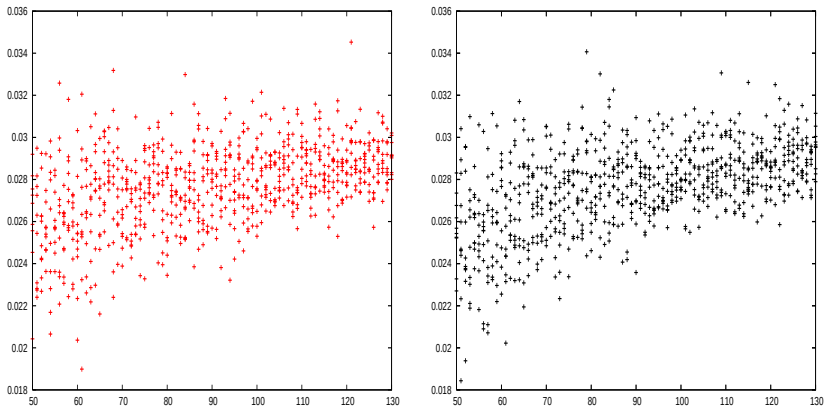

Fig. 3. Variation of $\frac{1}{d}\log\frac{\|\mathbf{b}_1\|}{(\det L)^{1/d}}$ as a function of d.

4.1 A Few Experiments

In Figure 3, we consider the variations of the quantity $\frac{1}{d}\log\frac{\|\mathbf{b}_1\|}{(\det L)^{1/d}}$ as the dimension d increases. On the left side of the figure, each point is a sample of the

following experiment: generate a random knapsack-type basis with $B = 2^{100 \cdot d}$ and reduce it with L^2 (the fast variant of `fplll-1.2` with $(\delta, \eta) = (0.999, 0.501)$). The points on the right side correspond to the same experiments, but starting with Ajtai-type bases, with $\alpha = 1.2$. The two sides of Figure 3 are similar and the quantity $\frac{1}{d} \log \frac{\|\mathbf{b}_1\|}{(\det L)^{1/d}}$ seems to converge slightly below 0.03 (the corresponding worst-case constant is ≈ 0.10). This means that the first output vector $\mathbf{b}_1$ usually satisfies $\|\mathbf{b}_1\| \approx (1.02)^d (\det L)^{1/d}$. The exponential quantity $(1.02)^d$ remains tiny even in moderate dimensions: e.g., $(1.02)^{50} \approx 2.7$ and $(1.02)^{100} \approx 7.2$. These data may explain why in the 80's, cryptanalysts used to believe that L^3 returns vectors surprisingly small compared to the worst-case bound.

4.2 The Configuration of Local Bases

To understand the shape of the bases that are computed by L^3, it is tempting to consider the local bases of the output bases, i.e., the pairs $(\mathbf{b}_i^*, \mu_{i+1,i}\mathbf{b}_i^* + \mathbf{b}_{i+1}^*)$ for $i < d$. These pairs are the components of $\mathbf{b}_i$ and $\mathbf{b}_{i+1}$ which are orthogonal to $\mathbf{b}_1, \ldots, \mathbf{b}_{i-1}$. We experimentally observe that after the reduction, local bases seem to share a common configuration, independently of the index i. In Figure 4, a point corresponds to a local basis (its coordinates are $\mu_{i+1,i}$ and $\|\mathbf{b}_{i+1}^*\|/\|\mathbf{b}_i^*\|$) of a basis returned by the fast variant of `fplll-1.2` with parameters $\delta = 0.999$ and $\eta = 0.501$, starting from a knapsack-type basis with $B = 2^{100 \cdot d}$. The 2100 points correspond to 30 reduced bases of 71-dimensional lattices. This distribution seems to stabilize between the dimensions 40 and 50.

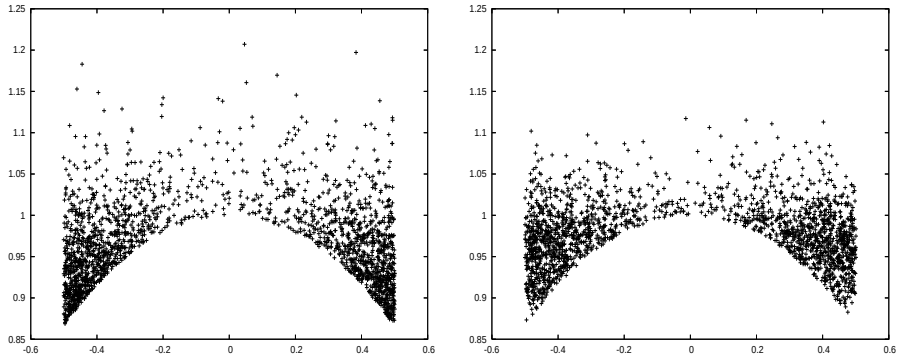

Fig. 4. Distribution of the local bases after L^3 (left) and deep-L^3 (right).

Figure 4 is puzzling. First of all, the $\mu_{i+1,i}$'s are not uniformly distributed in $[-\eta, \eta]$, as one may have thought *a priori*. As an example, the uniform distribution was used as an hypothesis Theorem 2 in [17]. Our observation therefore invalidates this result. This non-uniformity is surprising because the other $\mu_{i,j}$'s seem to be uniformly distributed in $[-\eta, \eta]$, in particular when $i - j$ becomes larger, as it is illustrated by Figure 5. The mean value of the $|\mu_{i+1,i}|$'s is close to 0.38. Besides, the mean value of $\|\mathbf{b}_i^*\|/\|\mathbf{b}_{i+1}^*\|$ is close to 1.04, which matches the 1.02 constant of the previous subsection. Indeed, if the local bases behave independently, we have:

$$\frac{\|\mathbf{b}_1\|^d}{\det L} = \prod_{i=1}^{d} \frac{\|\mathbf{b}_1\|}{\|\mathbf{b}_i^*\|} = \prod_{i=1}^{d-1} \left(\frac{\|\mathbf{b}_i^*\|}{\|\mathbf{b}_{i+1}^*\|} \right)^{d-i} \approx (1.04)^{d^2/2} \approx (1.02)^{d^2}.$$

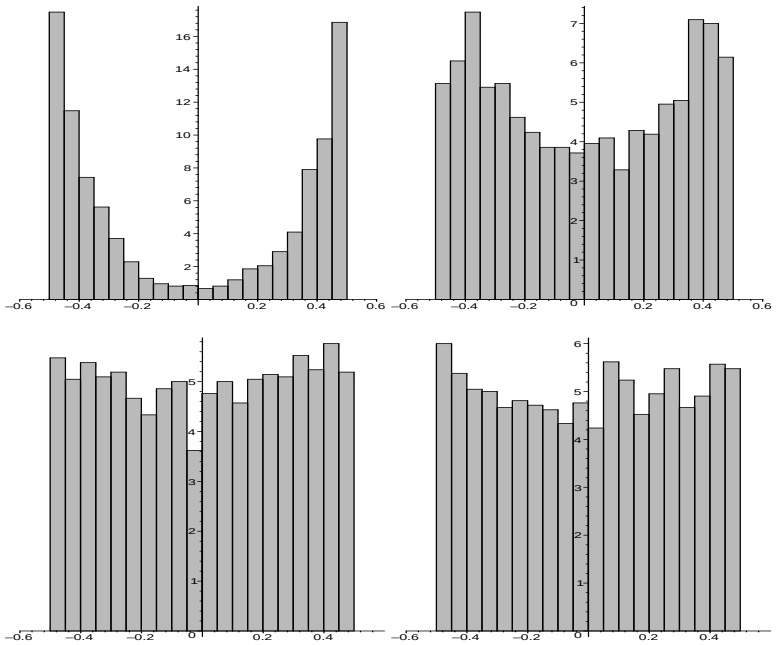

Fig. 5. Distribution of $\mu_{i,i-1}$ (top left), $\mu_{i,i-2}$ (top right), $\mu_{i,i-5}$ (bottom left) and $\mu_{i,i-10}$ (bottom right) for 71-dimensional lattices after L^3.

A possible explanation of the shape of the pairs $(\mathbf{b}_i^*, \mu_{i+1,i}\mathbf{b}_i^* + \mathbf{b}_{i+1}^*)$ is as follows. During the execution of L^3, the ratios $\|\mathbf{b}_i^*\|/\|\mathbf{b}_{i+1}^*\|$ are decreasing steadily. At some moment, the ratio $\|\mathbf{b}_i^*\|/\|\mathbf{b}_{i+1}^*\|$ becomes smaller than $\sqrt{4/3}$. When it does happen, relatively to $\mathbf{b}_i^*$, either $\mu_{i+1,i}\mathbf{b}_i^* + \mathbf{b}_{i+1}^*$ lies in one of the corners of Figure 4 or is close to the vertical axis. In the first case, it does not change since $(\mathbf{b}_i^*, \mu_{i+1,i}\mathbf{b}_i^* + \mathbf{b}_{i+1}^*)$ is reduced. Otherwise $\mathbf{b}_i$ and $\mathbf{b}_{i+1}$ are to be swapped since $\mu_{i+1,i}\mathbf{b}_i^* + \mathbf{b}_{i+1}^*$ is not in the fundamental domain.

4.3 Schnorr-Euchner's Deep Insertion

The study of local bases helps to understand the behavior of the Schnorr-Euchner deep insertion algorithm [29]. In deep-L^3, instead of having the Lovász conditions satisfied for the pairs $(i, i+1)$, one requires that they are satisfied for all pairs (i, j) with $i < j$, i.e.:

$$\|\mathbf{b}_j^* + \mu_{j,j-1}\mathbf{b}_{j-1}^* + \ldots + \mu_{j,i}\mathbf{b}_i^*\|^2 \geq \delta\|\mathbf{b}_i^*\|^2 \text{ for } j > i.$$

This is stronger than the L^3-reduction, but no polynomial-time algorithm to compute it is known. Yet in practice, if we deep-L^3-reduce an already L^3-reduced basis and if the dimension is not too high, it terminates reasonably fast. On the right side of Figure 4, we did the same experiment as on the left side, except that instead of only L^3-reducing the bases, we L^3-reduced them and then deep-L^3-reduced the obtained bases. The average value of $\|\mathbf{b}_i^*\|/\|\mathbf{b}_{i+1}^*\|$'s is closer to 1 than in the case of L^3: the 1.04 and 1.02 constants become respectively ≈ 1.025 and 1.012. These data match the observations of [6].

We explain this phenomenon as follows. Assume that, relatively to $\mathbf{b}_i^*$, the vector $\mu_{i+1,i}\mathbf{b}_i^* + \mathbf{b}_{i+1}^*$ lies in a corner in the left side of Figure 4. Then the Lovász condition between $\mathbf{b}_i$ and $\mathbf{b}_{i+2}$ is less likely to be fulfilled, and the vector $\mathbf{b}_{i+2}$ is more likely to be changed. Indeed, the component of $\mathbf{b}_{i+2}$ onto $\mathbf{b}_{i+1}^*$ will be smaller than usual (because $\|\mathbf{b}_{i+1}^*\|/\|\mathbf{b}_i^*\|$ is small), and thus $\mu_{i+2,i+1}\mathbf{b}_{i+1}^*$ will be smaller. As a consequence, the vector $\mu_{i+2,i}\mathbf{b}_i^* + \mu_{i+2,i+1}\mathbf{b}_{i+1}^* + \mathbf{b}_{i+2}^*$ is more likely to be shorter than $\mathbf{b}_i^*$, and thus $\mathbf{b}_{i+2}$ is more likely to change. Since the corner local bases arise with high frequency, deep-L^3 often performs insertions of depths higher than 2 that would not be performed by L^3.

5 The Practical Running Time of L^3

In this section, we argue that the worst case complexity bound $O(d^4(d+n)(d+\log B)\log B)$ is asymptotically reached for some classes of random bases, and explain how and why the running time is better in practice. Here we consider bases for which $n = \Theta(d) = O(\log B)$, so that the bound above becomes $O(d^5 \log^2 B)$. Notice that the heuristic codes do not have any asymptotic meaning since they do not terminate when the dimension increases too much (in particular, the working precision must increase with the dimension). Therefore, all the experiments described in this section were performed using the proved variant of `fplll-1.2`.

We draw below a heuristic worst-case complexity analysis of L^2 that will help us to explain the difference between the worst case and the practical behavior:

- There are $O(d^2 \log B)$ loop iterations.
- In a given loop iteration, there are usually two iterations within the lazy size-reduction: the first one makes the $|\mu_{\kappa,i}|$'s smaller than η and the second one recomputes the $\mu_{\kappa,i}$'s and $r_{\kappa,i}$'s with better accuracy. This is incorrect in full generality (in particular for knapsack-type bases as we will see below), but is the case most often.
- In each iteration of the size-reduction, there are $O(d^2)$ arithmetic operations.
- Among these, the most expensive ones are those related to the coefficients of the basis and Gram matrices: these are essentially multiplications between integers of lengths $O(\log B)$ and the x_i's, of lengths $O(d)$.

We argue that the analysis above is tight for Ajtai-type random bases.

Heuristic 2 *Let $\alpha > 1$. When d grows to infinity, the average cost of the L^2 algorithm given as input a randomly and uniformly chosen d-dimensional Ajtai-type basis with parameter α is $\Theta(d^{5+2\alpha})$.*

In this section, we also claim that the bounds of the heuristic worst-case analysis are tight in practice for Ajtai-type random bases, except the $O(d)$ bound on size of the x_i's. Finally, we detail the case of knapsack-type random bases.

5.1 L^2 on Ajtai-Type Random Bases

Firstly, the $O(d^2 \log B)$ bound on the loop iterations seems to be tight in practice, as suggested by Figure 6. The left side corresponds to Ajtai-type random bases with $\alpha = 1.2$: the points are the experimental data and the continuous line is

the `gnuplot` interpolation of the type $f(d) = a \cdot d^{3.2}$ (we have $\log B = O(d^{1.2})$). The right side has been obtained similarly, for $\alpha = 1.5$, and $g(d) = b \cdot d^{3.5}$. With Ajtai-type bases, size-reductions contain extremely rarely more than two iterations. For example, for $d \leq 75$ and $\alpha = 1.5$, fewer than 0.01% of the size-reductions involve more than two iterations. The third bound of the heuristic worst case analysis is also reached.

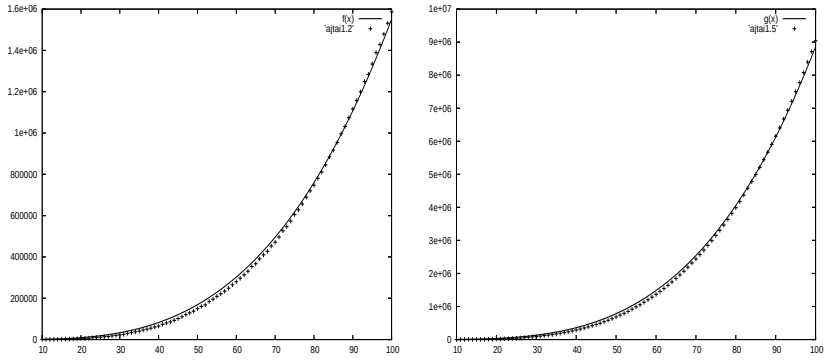

Fig. 6. Number of loop iterations of L^2 as a function of d, for Ajtai-type random bases.

These similarities between the worst and average cases do not go on for the size of the integers involved in the arithmetic operations. The x_i's computed during the size-reductions are most often shorter than a machine word, which makes it difficult to observe the $O(d)$ factor in the complexity bound coming from them. For an Ajtai-type basis with $d \leq 75$ and $\alpha = 1.5$, fewer than 0.2% of the non-zero x_i's are longer than 64 bits. In the worst case [23], we have $|x_i| \lesssim (3/2)^{\kappa-i} M$, where M is the maximum of the $\mu_{\kappa,j}$'s before the lazy size-reduction starts, and κ is the current L^3 index. In practice, M happens to be small most of the time. We argue that the average situation is $|x_i| \approx (1.04)^{\kappa-i} M$. This bound remains exponential, but for a small M, x_i becomes larger than a machine word only in dimensions higher than several hundreds. We define:

$$\mu_{\kappa,i}^{(\mathrm{final})} = \mu_{\kappa,i}^{(\mathrm{initial})} - \sum_{j=i+1}^{\kappa-1} x_j \mu_{j,i} = \mu_{\kappa,i}^{(\mathrm{initial})} - \sum_{j=i+1}^{\kappa-1} \left\lfloor \mu_{\kappa,j}^{(\mathrm{final})} \right\rceil \mu_{j,i}.$$

We model the $\left(\mu_{\kappa,\kappa-i}^{(\mathrm{final})} \right)_i$'s by the random variables U_i defined as follows:

$$U_0 = R_0 \quad \text{and} \quad U_i = R_i + \sum_{j=1}^{i-1} U_j R'_{i,j} \quad \text{if} \quad i \geq 1,$$

where the R_i's and $R'_{i,j}$'s are uniformly distributed respectively in $[-a, a]$ for some constant a and in $[-\eta, \eta]$. We assume that the R_i's and $R'_{i,j}$'s are pairwise independent. These hypotheses on the $\mu_{i,j}$'s are strong. In particular we saw in Section 4 that the $\mu_{i,i-1}$'s are not uniformly distributed in $[-\eta, \eta]$. Nevertheless, this simplification does not significantly change the asymptotic behavior of the sequence (U_i) and simplifies the technicalities. Besides, to make the model closer to the reality, we could have rounded the U_j's, but since these quantities are growing to infinity, this should not change much the asymptotic behavior. The independence of the R_i's and $R'_{i,j}$'s and their symmetry give:

$$\mathbb{E}\left[U_i\right] = 0, \ \mathbb{E}\left[U_i^2\right] = \mathbb{E}\left[R_i^2\right] + \sum_{j=1}^{i-1} \mathbb{E}\left[U_j^2\right] \cdot \mathbb{E}\left[R_{i,j}'^2\right] = \frac{2a^3}{3} + \frac{2\eta^3}{3} \sum_{j=1}^{i-1} \mathbb{E}\left[U_j^2\right].$$

As a consequence, for i growing to infinity, we have $\mathbb{E}[U_i^2] \approx \left(\frac{2\eta^3}{3}+1\right)^i$. If we choose $\eta \approx 1/2$, we get $\frac{2\eta^3}{3}+1 \approx \frac{13}{12} \approx 1.08$. We thus expect the $|x_i|$'s to be of length $\lesssim (\log_2 1.04)\cdot d \approx 0.057 \cdot d$. To sum up, the x_i's should have length $O(d)$ in practice, but the $O(\cdot)$-constant is tiny. For example, the quantity $(1.04)^d$ becomes larger than 2^{64} for $d \geq 1100$. Since we cannot reduce lattice bases which simultaneously have this dimension and reach the other bounds of the heuristic worst-case complexity analysis, it is at the moment impossible to observe the asymptotic behavior. The practical running time is rather to $O(d^4 \log^2 B)$.

5.2 L^2 on Knapsack-Type Bases

In the case of knapsack-type bases there are fewer loop iterations than in the worst case: the quantity $\Delta = \prod_{i=1}^{d} \|\mathbf{b}_i^*\|^{2(d-i+1)}$ of L^3's analysis satisfies $\Delta = B^{O(d)}$ instead of $\Delta = B^{O(d^2)}$. This ensures that there are $O(d \log B)$ loop iterations, so that the overall cost of L^2 for these lattice bases is $O(d^4 \log^2 B)$. Here we argue that asymptotically one should expect a better complexity bound.

Heuristic 3 *When d and $\log B$ grow to infinity with $\log B = \Omega(d^2)$, the average cost of the L^2 algorithm given as input a randomly and uniformly chosen d-dimensional knapsack-type basis with entries of length $\leq \log B$ is $\Theta(d^3 \log^2 B)$.*

In practice for moderate dimensions, the phenomenon described in the previous subsection makes the cost even lower: close to $O(d^2 \log^2 B)$ when $\log B$ is significantly larger than d.

First, there are $\Theta(d \log B)$ main loop iterations. These iterations are equally distributed among the different values of $\kappa_{\max}$: we define κ_{max} as the maximum of the indices κ since the beginning of the execution of the algorithm, i.e., the number of basis vectors that have been considered so far. We have $\kappa_{\max} = 2$ at the beginning, then $\kappa_{\max}$ is gradually incremented up to $d+1$, when the execution of L^2 is over. The number of iterations for each $\kappa_{\max}$ is roughly the same, approximately $\Theta(\log B)$. We divide the execution into $d-1$ phases, according to the value of $\kappa_{\max}$. We observe experimentally that at the end of the phase of a given $\kappa_{\max}$, the current basis has the following shape:

$$\begin{pmatrix}
a_{1,1} & a_{1,2} & \dots & a_{1,\kappa_{\max}+1} & 0 & 0 & \dots & 0 \\
a_{2,1} & a_{2,2} & \dots & a_{2,\kappa_{\max}+1} & 0 & 0 & \dots & 0 \\
\vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\
a_{\kappa_{\max},1} & a_{\kappa_{\max},2} & \dots & a_{\kappa_{\max},\kappa_{\max}+1} & 0 & 0 & \dots & 0 \\
A_{\kappa_{\max}+1} & 0 & \dots & 0 & 1 & 0 & \dots & 0 \\
A_{\kappa_{\max}+2} & 0 & \dots & 0 & 0 & 1 & \dots & 0 \\
\vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\
A_d & 0 & \dots & 0 & 0 & 0 & \dots & 1
\end{pmatrix},$$

where the top left $a_{i,j}$'s satisfy: $|a_{i,j}| = O\left(B^{\frac{1}{\kappa_{\max}}}\right)$.

We subdivide each $\kappa_{\max}$-phase in two subphases: the first subphase is the first loop iteration of L^2 for which $\kappa = \kappa_{\max}$, and the second one is made of the other iterations with the same $\kappa_{\max}$. The first subphase shortens the vector $\mathbf{b}_{\kappa_{\max}}$:

its length decreases from $\approx B$ to $\leq \sqrt{\kappa_{\max}(\max_{i<\kappa_{\max}} \|\mathbf{b}_i\|^2)+1} \lesssim B^{\frac{1}{\kappa_{\max}-1}}$. This subphase costs $O(d\log^2 B)$ bit operations (see [23]): there are $O(\log B/d)$ loop iterations in the lazy size-reduction; each one involves $O(d^2)$ arithmetic operations; among them, the most costly are the integer multiplications between the x_i's (that are $O(d)$-bit long) and the coefficients of the basis and Gram matrices (their lengths are $O(\log B/d)$, except the $\langle \mathbf{b}_\kappa, \mathbf{b}_i\rangle$'s which occur with frequency $1/O(\kappa)$). The global cost of the first subphases is $O(d^2\log^2 B)$. This is negligible in comparison to the overall cost of the second subphases.

Let $\mathbf{b}'_i$ be the vector obtained from $\mathbf{b}_i$ after the first subphase of the phase for which $\kappa_{\max} = i$, that is, right after its first size-reduction. Let $C(d, B)$ be the overall cost of the second subphases in dimension d and for input A_i's satisfying $|A_i| \leq B$. We divide the execution of the algorithm as follows: it starts by reducing a knapsack-type basis of dimension $\lfloor d/2 \rfloor$; let $\left(\mathbf{b}''_1, \ldots, \mathbf{b}''_{\lfloor d/2\rfloor}\right)$ be the corresponding L^3-reduced vectors; if we exclude the $\lceil d/2\rceil$ remaining first subphases, then L^2 reduces the basis $\left(\mathbf{b}''_1, \ldots, \mathbf{b}''_{\lfloor d/2\rfloor}, \mathbf{b}'_{\lfloor d/2+1\rfloor}, \ldots, \mathbf{b}'_d\right)$, where all the lengths of the vectors are bounded by $\approx B^{2/d}$. As a consequence, we have:

$$C(d, B) = C(d/2, B) + O(d^5(\log B/d)^2) = C(d/2, B) + O(d^3\log^2 B),$$

from which one easily gets $C(d, B) = O(d^3\log^2 B)$, as long as $d^2 = O(\log B)$.

5.3 Many Parameters Can Influence the Running Time

We list below a few tunings that should be performed if one wants to optimize L^3 and L^2 for particular instances:

- Firstly, use as less multiprecision arithmetic as possible. If you are in a medium dimension (e.g., less than 170), you may avoid multiprecision *fpa* (see Section 6). If your input basis is made of short vectors, like for NTRU lattices, try using chip integers instead of multiprecision integers.
- Detect if there are scalar products cancellations: if these cancellations happen very rarely, use a heuristic variant that does not require the Gram matrix. Otherwise, if such cancellations happen frequently, a proved variant using the Gram matrix may turn out to be cheaper than a heuristic one recomputing exactly many scalar products.
- It is sometimes recommended to weaken δ and η. Indeed, if you increase η and/or decrease δ, it will possibly decrease the number of iterations within the lazy size-reduction and the number of global iterations. However, relaxed L^3-factors require a higher precision: for a given precision, the dimension above which L^2 might loop forever decreases (see Section 6).

6 "Numerical Stability" of L^3

In this section, we discuss problems that may arise when one uses *fpa* within L^3. The motivation is to get a good understanding of the "standard" numerical behavior, in order to keep the double precision as long as possible with low chances of failure. Essentially, two different phenomena may be encountered: a lazy size-reduction or consecutive Lovász tests may be looping forever. The output may

also be incorrect, but most often if something goes wrong, the execution loops within a size-reduction. We suppose here that either the Gram matrix is maintained exactly during the execution or that the problems arising from scalar product cancellations do not show up.

It is shown in [23] that for some given parameters δ and η, a precision of $\left(\log \frac{(1+\eta)^2}{\delta-\eta^2} + \varepsilon\right) \cdot d + o(d)$ is sufficient for L^2 to work correctly, for any constant $\varepsilon > 0$. For δ close to 1 and η close to $1/2$, it gives that a precision of $1.6 \cdot d + o(d)$ suffices. A family of lattice bases for which this bound seems to be tight is also given. Nevertheless, in practice the algorithm seems to work correctly with a much lower precision: for example, the double precision (53 bit-long mantissæ) seems sufficient most of the time up to dimension 180. We argue here that the average required precision grows linearly with the dimension, but with a significantly lower constant.

Heuristic 4 *Let δ be close to 1 and η be close to $1/2$. For almost every lattice, with a precision of $0.25 \cdot d + o(d)$ bits for the* fp*-calculations, the L^2 algorithm performs correctly when given almost any input basis.*

This heuristic has direct consequences for a practical implementation of L^2: it helps guessing what precision should be sufficient in a given dimension, and thus a significant constant factor can be saved for the running time.

We now give a justification for the heuristic above. For a fixed size of mantissa, we evaluate the dimension for which things should start going wrong. First, we evaluate the error made on the Gram-Schmidt coefficients and then we will use these results for the behavior of L^3: to do this, we will say that L^3 performs plenty of Gram-Schmidt calculations (during the successive loop iterations), and that things go wrong if at least one of these calculations is erroneous.

We consider the following random model, which is a simplified version of the one described in Section 4 (the simplification should not change the asymptotic results but helps for the analysis).

- The $\mu_{i,j}$'s for $i > j$ are chosen randomly and independently in $[-\eta, \eta]$. They share a distribution that is symmetric towards 0. This implies that $\mathbb{E}[\mu] = 0$. We define $\mu_2 = \mathbb{E}[\mu^2]$ and $\mu_{i,i} = 1$.
- The $\frac{r_{i,i}}{r_{i+1,i+1}}$'s are chosen randomly and independently in $(0, \beta]$. These choices are independent of those of the $\mu_{i,j}$'s. We define $\alpha = \mathbb{E}\left[\frac{r_{i,i}}{r_{i+1,i+1}}\right]$.
- The random variables $\mu_{i,j}$ and $\frac{r_{i,i}}{r_{i+1,i+1}}$ determine the Gram matrix of the initial basis. Let $r_{1,1}$ be an arbitrary constant. We define the following random variables, for $i \geq j$: $\langle \mathbf{b}_i, \mathbf{b}_j \rangle = r_{1,1} \sum_{k=1}^{j} \mu_{j,k}\mu_{i,k} \prod_{l=1}^{k-1} (r_{l,l}/r_{l+1,l+1})^{-1}$.
- We define the random variables $r_{i,j} = r_{1,1}\mu_{i,j} \prod_{l=1}^{j-1} (r_{l,l}/r_{l+1,l+1})^{-1}$ (for $i \geq j$).
- We assume that we do a relative error $\varepsilon = 2^{-\ell}$ (with ℓ the working precision) while translating the exact value $\|\mathbf{b}_1\|^2$ into a *fp* number: $\Delta\|\mathbf{b}_1\|^2 = \varepsilon\|\mathbf{b}_1\|^2$.

We have selected a way to randomly choose the Gram matrix and to perform a rounding error on $\|\mathbf{b}_1\|^2$. To simplify the analysis, we suppose that there is no rounding error performed on the other $\langle \mathbf{b}_i, \mathbf{b}_j \rangle$'s. Our goal is to estimate

the amplification of the rounding error $\Delta\|\mathbf{b}_1\|^2$ during the calculations of approximations of the $r_{i,j}$'s and $\mu_{i,j}$'s. We neglect high-order error terms. More precisely, we study the following random variables, defined recursively:

$$\Delta r_{1,1} = \Delta\|\mathbf{b}_1\|^2 = \varepsilon\|\mathbf{b}_1\|^2,$$
$$\Delta r_{i,j} = -\sum_{k=1}^{j-1}(\Delta r_{i,k}\mu_{j,k} + \Delta r_{j,k}\mu_{i,k} - \Delta r_{k,k}\mu_{i,k}\mu_{j,k}) \text{ when } i \geq j,$$

The $\mu_{a,b}$'s and $\frac{r_{b,b}}{r_{b+1,b+1}}$'s that may not be independent with $\Delta r_{i,k}$ are those for which $b < k$. As a consequence, $\Delta r_{i,k}, \mu_{j,k}, \frac{r_{j-1,j-1}}{r_{j,j}}, \frac{r_{j-2,j-2}}{r_{j-1,j-1}}, \ldots, \frac{r_{k,k}}{r_{k+1,k+1}}$ are pairwise independent, $\Delta r_{j,k}, \mu_{i,k}, \frac{r_{j-1,j-1}}{r_{j,j}}, \frac{r_{j-2,j-2}}{r_{j-1,j-1}}, \ldots, \frac{r_{k,k}}{r_{k+1,k+1}}$ are pairwise independent, and $\Delta r_{k,k}, \mu_{i,k}, \mu_{j,k}, \frac{r_{j-1,j-1}}{r_{j,j}}, \frac{r_{j-2,j-2}}{r_{j-1,j-1}}, \ldots, \frac{r_{k,k}}{r_{k+1,k+1}}$ are pairwise independent, for all (i,j,k) satisfying $i > j > k$. Therefore, for any $i > j$:

$$\mathbb{E}\left[\frac{\Delta r_{i,j}}{r_{j,j}}\right] = -\sum_{k=1}^{j-1}\left(\prod_{l=k}^{j-1}\mathbb{E}\left[\frac{r_{l,l}}{r_{l+1,l+1}}\right]\right)\left(\mathbb{E}\left[\frac{\Delta r_{i,k}}{r_{k,k}}\right]\mathbb{E}[\mu_{j,k}] + \mathbb{E}\left[\frac{\Delta r_{j,k}}{r_{k,k}}\right]\mathbb{E}[\mu_{i,k}] - \mathbb{E}\left[\frac{\Delta r_{k,k}}{r_{k,k}}\right]\mathbb{E}[\mu_{i,k}]\mathbb{E}[\mu_{j,k}]\right).$$

Because $\mathbb{E}[\mu_{j,k}] = \mathbb{E}[\mu_{i,k}] = 0$, we get $\mathbb{E}\left[\frac{\Delta r_{i,j}}{r_{j,j}}\right] = 0$, for all $i > j$. Similarly, we have, for $i > 1$:

$$\mathbb{E}\left[\frac{\Delta r_{i,i}}{r_{i,i}}\right] = \mu_2\sum_{k=1}^{j-1}\left(\prod_{l=k}^{i-1}\mathbb{E}\left[\frac{r_{l,l}}{r_{l+1,l+1}}\right]\right)\mathbb{E}\left[\frac{\Delta r_{k,k}}{r_{k,k}}\right] = \mu_2\sum_{k=1}^{j-1}\alpha^{i-k}\mathbb{E}\left[\frac{\Delta r_{k,k}}{r_{k,k}}\right].$$

We obtain that $\mathbb{E}\left[\frac{\Delta r_{i,i}}{r_{i,i}}\right]$ is close to $(\alpha(1+\mu_2))^i\varepsilon$. For example, if the $\mu_{i,j}$'s are uniformly chosen in $[-1/2, 1/2]$, if $\alpha = 1.08$ (as observed in Section 4), and if $\varepsilon \approx 2^{-53}$, we get $\mathbb{E}\left[\frac{\Delta r_{i,i}}{r_{i,i}}\right] \approx 1.17^i \cdot 2^{-53}$. For $i = 180$, this is close to 2^{-12}.

We have analyzed very roughly the influence of the rounding error made on $\|\mathbf{b}_1\|^2$, within the Gram-Schmidt orthogonalization for L^3-reduced bases. If we want to adapt this analysis to L^2, we must take into account the number of $r_{i,j}$'s and $\mu_{i,j}$'s that are computed during the execution. To simplify we consider only the $r_{d,d}$'s, which are *a priori* the less accurate. We suppose that all the computations of $r_{d,d}$ through the execution are independent. Let K be the number of iterations for which $\kappa = d$. We consider that an error on $r_{d,d}$ is significant if $\frac{\Delta r_{d,d}}{r_{d,d}}$ is at least 2^{-3}. If such an error occurs, the corresponding Lovász test is likely to be erroneous. Under such hypotheses, the probability of failure is of the order of $1-(1-2^{-12+3})^K \approx K2^{-9}$. In case of several millions of Lovász tests, it is very likely that there is one making L^3 behave unexpectedly.

The above analysis is completely heuristic and relies on very strong hypotheses, but it provides orders of magnitude that one seems to encounter in practice. For random bases, we observe infinite loops in double precision arising around dimensions 175 to 185, when there are a few millions Lovász tests.

Acknowledgments. We thank Guillaume Hanrot for helpful discussions. The writing of this paper was completed while the second author was visiting the University of Sydney, whose hospitality is gratefully acknowledged.

References

1. M. Ajtai. Generating hard instances of lattice problems (extended abstract). In *Proc. of STOC 1996*, pages 99–108. ACM, 1996.
2. M. Ajtai. Random lattices and a conjectured 0-1 law about their polynomial time computable properties. In *Proc. of FOCS 2002*, pages 13–39. IEEE, 2002.
3. M. Ajtai. The worst-case behavior of Schnorr's algorithm approximating the shortest nonzero vector in a lattice. In *Proc. of STOC 2003*, pages 396–406. ACM, 2003.
4. M. Ajtai. Generating Random Lattices According to the Invariant Distribution. Draft, 2006.
5. L. Babai. On Lovász lattice reduction and the nearest lattice point problem. *Combinatorica*, 6:1–13, 1986.
6. W. Backes and S. Wetzel. Heuristics on lattice reduction in practice. *ACM Journal of Experimental Algorithms*, 7:1, 2002.
7. C. Batut, K. Belabas, D. Bernardi, H. Cohen, and M. Olivier. PARI/GP computer package version 2. Av. at `http://pari.math.u-bordeaux.fr/`.
8. J. W. S. Cassels. *Rational quadratic forms*, volume 13 of *London Mathematical Society Monographs*. Academic Press Inc. [Harcourt Brace Jovanovich Publishers], London, 1978.
9. H. Cohen. *A Course in Computational Algebraic Number Theory, 2nd edition*. Springer-V., 1995.
10. D. Coppersmith. Small solutions to polynomial equations, and low exponent RSA vulnerabilities. *Journal of Cryptology*, 10(4):233–260, 1997.
11. O. Goldreich, S. Goldwasser, and S. Halevi. Public-key cryptosystems from lattice reduction problems. In *Proc. of Crypto 1997*, volume 1294 of *LNCS*, pages 112–131. Springer-V., 1997.
12. D. Goldstein and A. Mayer. On the equidistribution of Hecke points. *Forum Mathematicum*, 15:165–189, 2003.
13. G. Golub and C. van Loan. *Matrix Computations*. J. Hopkins Univ. Press, 1996.
14. L. Groetschel, L. Lovász, and A. Schrijver. *Geometric Algorithms and Combinatorial Optimization*. Springer-V., 1988.
15. C. Hermite. Extraits de lettres de M. Hermite à M. Jacobi sur différents objets de la théorie des nombres, deuxième lettre. *Journal für die reine und angewandte Mathematik*, 40:279–290, 1850.
16. J. Hoffstein, J. Pipher, and J. H. Silverman. NTRU: a ring based public key cryptosystem. In *Proc. of ANTS III*, volume 1423 of *LNCS*, pages 267–288. Springer-V., 1998.
17. H. Koy and C. P. Schnorr. Segment LLL-reduction of lattice bases with floating-point orthogonalization. In *Proc. of CALC '01*, volume 2146 of *LNCS*, pages 81–96. Springer-V., 2001.
18. A. K. Lenstra, H. W. Lenstra, Jr., and L. Lovász. Factoring polynomials with rational coefficients. *Mathematische Annalen*, 261:513–534, 1982.
19. H. W. Lenstra, Jr. Integer programming with a fixed number of variables. Technical report 81-03, Mathematisch Instituut, Universiteit van Amsterdam, 1981.
20. H. W. Lenstra, Jr. Integer programming with a fixed number of variables. *Mathematics of Operations Research*, 8(4):538–548, 1983.
21. Magma. The Magma computational algebra system for algebra, number theory and geometry. Av. at `http://www.maths.usyd.edu.au:8000/u/magma/`.
22. D. Micciancio and S. Goldwasser. *Complexity of lattice problems: a cryptographic perspective*. Kluwer Academic Press, 2002.
23. P. Q. Nguyen and D. Stehlé. Floating-point LLL revisited. In *Proc. of Eurocrypt 2005*, volume 3494 of *LNCS*, pages 215–233. Springer-V., 2005.
24. P. Q. Nguyen and J. Stern. The two faces of lattices in cryptology. In *Proc. of CALC '01*, volume 2146 of *LNCS*, pages 146–180. Springer-V., 2001.
25. A. M. Odlyzko. The rise and fall of knapsack cryptosystems. In *Proc. of Cryptology and Computational Number Theory*, volume 42 of *Proc. of Symposia in Applied Mathematics*, pages 75–88. AMS, 1989.
26. The SPACES Project. MPFR, a LGPL-library for multiple-precision floating-point computations with exact rounding. Av. at `http://www.mpfr.org/`.
27. C. P. Schnorr. A hierarchy of polynomial lattice basis reduction algorithms. *Theoretical Computer Science*, 53:201–224, 1987.
28. C. P. Schnorr. A more efficient algorithm for lattice basis reduction. *Journal of Algorithms*, 9(1):47–62, 1988.
29. C. P. Schnorr and M. Euchner. Lattice basis reduction: improved practical algorithms and solving subset sum problems. *Mathematics of Programming*, 66:181–199, 1994.
30. V. Shoup. NTL, Number Theory C++ Library. Av. at `http://www.shoup.net/ntl/`.
31. C. L. Siegel. A mean value theorem in geometry of numbers. *Annals of Mathematics*, 46(2):340–347, 1945.