Worst–case complexity of the optimal LLL algorithm

ALI AKHAVI

GREYC - Université de Caen, F-14032 Caen Cedex, France
ali.akhavi@info.unicaen.fr

Abstract. In this paper, we consider the open problem of the complexity of the LLL algorithm in the case when the approximation parameter t of the algorithm has its extreme value 1. This case is of interest because the output is then the strongest Lovász–reduced basis. Experiments reported by Lagarias and Odlyzko [LO83] seem to show that the algorithm remain polynomial in average. However no bound better than a naive exponential order one is established for the worst–case complexity of the optimal LLL algorithm, even for fixed small dimension (higher than 2). Here we prove that, for any *fixed dimension* n, the number of iterations of the LLL algorithm is *linear* with respect to the size of the input. It is easy to deduce from [Val91] that the linear order is optimal. Moreover in 3 dimensions, we give a tight bound for the maximum number of iterations and we characterize precisely the output basis. Our bound also improves the known one for the usual (non–optimal) LLL algorithm.

1 Introduction

A Euclidean lattice is a set of all integer linear combinations of p linearly independent vectors in $\mathbb{R}^n$. The vector space $\mathbb{R}^n$ is then called *the ambient space.* Any lattice can be generated by many bases (all of them of cardinality p). The lattice basis reduction problem is to find bases with good Euclidean properties, that is sufficiently short vectors and almost orthogonal. The problem is old and there exist numerous notions of reduction; the most natural ones are due to Minkowski or to Korkhine–Zolotarev. For a general survey, see for example [Kan87,Val89]. Both of these reduction processes are "strong", since they build reduced bases with in some sense best Euclidean properties. However, they are also computationally hard to find, since they demand the first vector of the basis should be a shortest one in the lattice. It appears that finding such an element in a lattice is likely to be NP-hard [vEB81,Ajt97,Mic98,Cai99].

Fortunately, even approximate answers to the reduction problem have numerous theoretical and practical applications in computational number theory and cryptography: Factoring polynomials with rational coefficients [LLL82], finding linear Diophantine approximations [Lag80], breaking various cryptosystems [Lag83,Sch95,VGT88] and integer linear programming [Kan83,Len83]. In 1982, Lenstra, Lenstra and Lovász [LLL82] gave a powerful approximation reduction algorithm. It depends on a real approximation parameter $\delta \in]1, 2[$ and is called LLL(δ). It is a possible generalization of its 2–dimensional version, which is the famous Gauss algorithm. The celebrated LLL

algorithm seems difficult to analyze precisely, both in the worst–case and in average–case. The original paper [LLL82] gives an upper bound for the number of iterations of LLL(δ), which is polynomial in the data size, for all values of δ except the optimal value 1: When given n input vectors of $\mathbb{R}^n$ of length at most M, the data size is $\mathrm{O}(n^2 \log M)$ and the upper bound is $n^2 \log_\delta M + n$. When the approximation parameter δ is 1, the only known upper–bound is M^{n^2}, which is *exponential* even for fixed dimension. It was still an open problem whether the optimal LLL algorithm is polynomial. In this paper, we prove that the number of iterations of the algorithm is *linear* for any fixed dimension. More precisely, it is $\mathrm{O}(A^{n^3} \log M)$ where A is any constant strictly greater than $(2/\sqrt{3})^{(1/6)}$. We prove also that under a quite reasonable heuristic principle, the number of iterations is $\mathrm{O}((2/\sqrt{3})^{n^2/2} \log M)$. In the 3–dimensional case (notice that the problem was totally open even in this case), we provide a precise linear bound, which is even better than the usual bounds on the non–optimal versions of the LLL algorithm.
Several reasons motivate our work on the complexity of the optimal LLL algorithm.
1. This problem is cited as an open question by respected authors [BK84,Val91] and I think that the answer will bring at least a better understanding of the lattice reduction process. Of course, this paper is just an insight to the the general answer of the question.
2. The optimal LLL algorithm provides the strongest Lovász–reduced basis in a lattice (the best bounds on the classical length defects and orthogonality defect). In many applications, people seem to be interested by such a basis [LO83], and sometimes even in fixed low dimension [Sch87].
3. We believe that the complexity of finding an optimal Lovász–reduced basis is of great interest and the LLL algorithm is the most natural way to find an optimal Lovász–reduced basis in a lattice[1].

Plan of the paper. In Section 2, we recall what is the LLL algorithm and we give some definitions and notations. In Section 3, we recall some known results in 2 dimensions. Section 4 deals with the worst–case complexity of the optimal LLL algorithm in 3-dimensional case. Finally, in Section 5, we prove that in fixed dimension, the LLL algorithm is linear with respect to the length of the input.

2 General description of the LLL algorithm

Let $\mathbb{R}^p$ be endowed with the usual scalar product $(\ ,\ )$ and Euclidean length $|u| = (u,u)^{1/2}$. The notation $(u)_{\perp H}$ will denote the projection of the vector u in the classical orthogonal space $H^\perp$ of H in $\mathbb{R}^p$. The set $\langle u_1, u_2, ..., u_r\rangle$ denotes the vector space spanned by a family of vectors $(u_1, u_2, ..., u_r)$. A lattice of $\mathbb{R}^p$ is the set of all integer linear combinations of a set of linearly independent vectors. Generally it is given by one of its bases $(b_1, b_2, \ldots, b_n)$ and the number n is the dimension of the lattice. So, if M is the maximum length of the vectors b_i, the data-size is $(n^2 \log M)$ and when working in fixed dimension, the data-size is $\mathrm{O}(\log M)$. The determinant $\det(L)$ of the lattice L is the volume of the n–dimensional parallelepiped spanned by the origin and the vectors of any basis. Indeed it does not depend on the choice of a basis.

[1] This point is more developed in the conclusion

The usual Gram–Schmidt orthogonalization process, builds in polynomial–time from a basis $b = (b_1, b_2, \ldots, b_n)$ an orthogonal basis $b^* = (b_1^*, b_2^*, \ldots, b_n^*)$ and a lower–triangular matrix $m = (m_{ij})$ that expresses the system b into the system b^*[2]. By construction,

$$\begin{cases} b_1^* = b_1 \\ b_2^* = b_{2\perp\langle b_1\rangle} \\ \vdots \\ b_i^* = b_{i\perp\langle b_1,\cdots,b_{i-1}\rangle} \\ \vdots \\ b_n^* = b_{n\perp\langle b_1,\cdots,b_{n-1}\rangle} \end{cases}, \quad m = \begin{array}{c} \\ b_1 \\ b_2 \\ \vdots \\ b_i \\ b_{i+1} \\ \vdots \\ b_n \end{array} \begin{array}{c} \begin{array}{ccccccc} b_1^* & b_2^* & \cdots & b_i^* & b_{i+1}^* & \cdots & b_n^* \end{array} \\ \begin{pmatrix} 1 & 0 & \cdots & \cdots & \cdots & \cdots & 0 \\ m_{21} & 1 & \ddots & & & & \vdots \\ \vdots & \vdots & \ddots & \ddots & & & \vdots \\ \cdots & \cdots & \cdots & 1 & 0 & & \vdots \\ \cdots & \cdots & \cdots & m_{i+1,i} & 1 & \ddots & \vdots \\ \vdots & \vdots & \vdots & \vdots & & \ddots & 0 \\ m_{n1} & m_{n2} & \cdots & \cdots & \cdots & \cdots & 1 \end{pmatrix} \end{array}. \tag{1}$$

We recall that if L is the lattice generated by the basis b, then the determinant $\det(L)$ satisfies

$$\det(L) = \prod_{i=1}^{n} |b_i^*|. \tag{2}$$

The ordered basis b is called *proper* if $|m_{ij}| \leq 1/2$, for $1 \leq j < i \leq n$.
There exists a simple algorithm which makes any basis, proper in polynomial–time, by means of some adequate integer translation of each b_i in the directions of b_j, for j decreasing from $i-1$ to 1.

Definition 1. [LLL82] For a real parameter $\delta \in [1, 2[$, the basis $(b_1, \ldots, b_n)$ is called δ–reduced (or LLL(δ)-reduced or δ-Lovász–reduced) if it fullfils the two following conditions:

(i) $(b_1, ..., b_n)$ is proper.
(ii) $\forall i \in \{1, 2, \cdots, n-1\} \qquad (1/\delta)|b_{i\perp\langle b_1,\cdots,b_{i-1}\rangle}| < |\left(b_{(i+1)}\right)_{\perp\langle b_1,\cdots,b_{i-1}\rangle}|$.

The optimal LLL(1) algorithm is a possible generalization of its 2–dimensional version, which is nothing but the famous Gauss algorithm, whose precise analysis is already done both in the worst–case [Lag80,Val91,KS96] and in the average–case [DFV97]. In the sequel, a reduced basis denotes always a LLL(1)-reduced basis. When we talk about the algorithm without other precision, we always mean the optimal LLL algorithm.
We adopt the following notations for all integer i in $\{1, 2, \cdots, n-1\}$,

$$\begin{cases} u_i := b_{i\perp\langle b_1,\cdots,b_{i-1}\rangle} & = b_i^* \\ v_i := (b_{i+1})_{\perp\langle b_1,\cdots,b_{i-1}\rangle} & = m_{i+1,i} b_i^* + b_{i+1}^* \\ B_i := \text{ the two–dimensional basis } (u_i, v_i). \end{cases} \tag{3}$$

Then, by the previous Definition $(b_1, \ldots, b_n)$ is reduced iff it is proper and if all bases B_i are reduced (Gauss-reduced) for $i \in \{1, 2, \ldots, n-1\}$.

[2] Of course, b^* is generally not a basis for the lattice generated by b.

Definition 2. Let t be a real parameter such that $t > 1$. We call a basis $(b_1, \ldots, b_n)$ *t–quasi–reduced* if it satisfies the following conditions:

(i) the basis $(b_1, \ldots, b_{n-1})$ is proper.
(ii) For all $1 \leq i \leq n-2$, the bases B_i are reduced.
(iii) The last basis B_{n-1} is not reduced but it is t–reduced[3]

$$|m_{n,n-1}| < 1/2 \quad \text{and} \quad (1/t)|u_{n-1}| \ \leq \ |v_{n-1}| < |u_{n-1}|.$$

In other words, whenever the beginning basis $(b_1, \cdots, b_{n-1})$ is reduced, but the whole basis $b = (b_1, \cdots, b_n)$ is not, then for all $t > 1$ such that the last two–dimensional basis B_i is t–reduced, the basis b is called t–quasi-reduced.
Here is a simple enunciation of the LLL(δ) algorithm:

The LLL(δ)-reduction algorithm:

Input: A basis $b = (b_1, \ldots, b_n)$ of a lattice L.
Output: A $LLL(1)$-reduced basis b of the lattice L.
Initialization: Compute the orthogonalized system b^* and the matrix m.
$\mathbf{i := 1}$;
While $\mathbf{i < n}$ do
$b_{i+1} := b_{i+1} - \lfloor m_{i+1,i} \rceil b_i$ ($\lfloor x \rceil$ is the integer nearest to x).
Test: Is the two-dimensional basis B_i δ–reduced?
If true, make $(b_1, \ldots, b_{i+1})$ proper by translations; **set** $\mathbf{i := i + 1}$;
If false, swap b_i and b_{i+1}; update b^* and m; if $i \neq 1$ then **set** $\mathbf{i := i - 1}$;

During an execution of the algorithm, the index i variates in $\{1, \ldots, n\}$. It is called the *current index*. When i equals some $k \in \{1, \ldots, n-1\}$, the beginning lattice generated by $(b_1, \ldots, b_k)$ is already reduced. Then, the reduction of the basis B_k is tested. If the test is positive, the basis $(b_1, \ldots, b_{k+1})$ is made proper and the beginning lattice generated by $(b_1, \ldots, b_{k+1})$ is then reduced. So, i is incremented. Otherwise, the vectors b_k and b_{k+1} are swapped. At this moment, nothing guarantees that $(b_1, \ldots, b_k)$ "remains" reduced. So, i is decremented. The algorithm updates b^* and m, translates the new b_k in the direction of b_{k-1} and tests the reduction of the basis B_{k-1}. Thus, the index i may fall down to 1. Finally, when i equals n, the whole basis is reduced and the algorithm terminates. The variation of the index i during an example of execution of the LLL algorithm is shown by Figure 1.
In the sequel, an iteration of the LLL algorithm is precisely an iteration of the "while" loop, in the previous enunciation. Then, each iteration has exactly one test (Is the two–dimensional basis B_i reduced ?). So the number of steps is exactly the number of tests. Notice that whenever a test at a level i is negative, i.e. the basis B_i, is not reduced, after the swap of b_i and b_{i+1}, the determinant d_i of the lattice $(b_1, \ldots, b_i)$ is decreased. Moreover, for any $t > 1$, if at the moment of the test, the basis B_i is even not t–reduced, the determinant d_i is decreased by a factor at least $1/t$. This explains the following definition.

[3] The last basis B_i is also called t–Gauss reduced.

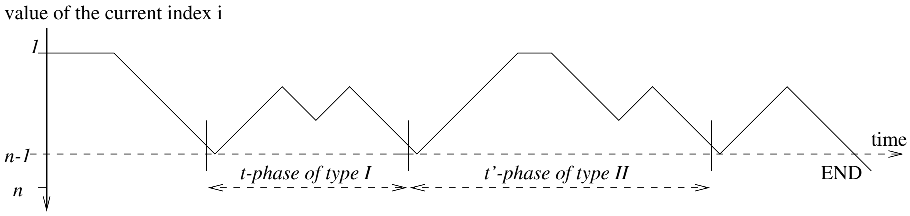

Fig. 1. Variation of the index i presented as a walk.

Definition 3. For a real parameter $t > 1$, a step of index i is called *t–decreasing* if at the moment of the test, the basis B_i is not t–reduced. Else, the step is called *t–non–decreasing*.

[LLL82] pointed out that during the execution of a non–optimal LLL algorithm, say LLL(δ) for some $\delta > 1$, all steps with negative tests are δ–decreasing. We use the same argument to show the following Lemma.

Lemma 4. *Let the LLL(1) algorithm work on an integer input basis $(b_1, \ldots, b_n)$ whose lengths of vectors is less than M. Let t be a real parameter such that $t > 1$. The total number of t–decreasing steps is always less than $n(n-1)/2 \log_t M$.*

Proof. The proof is based on the decrease of the integer quantity

$$D := \prod_{i=1}^{n-1} d_i^2 := \prod_{i=1}^{n-1} \prod_{j=1}^{i} |b_j^*|^2, \tag{4}$$

by the factor $(1/t^2)$, whenever a step is t–decreasing and the fact that other steps do not make D increase.

Definition 5. *A phase* is a sequence of iterations that occur between two successive tests of reduction of the last two–dimensional lattice B_{n-1}. For a real $t > 1$, we say that a phase is a *t–phase* if at the beginning of the phase, the basis $(b_1, \ldots b_n)$ is t–quasi–reduced. Moreover, phases are classified in two groups: A phase is called *of type I* if during the next phase, the first vector b_1 is never swapped. Else, it is called of *of type II*(see Figure 1).

3 Some known results in 2 dimensions: Gauss' algorithm

When working in two–dimensional space, each phase of the algorithm coincides with an iteration (an iteration of the "while" loop). Moreover, the only positive test occurs at the end of the algorithm. Thus, the number of steps is bounded from above by the maximum number of negative tests plus one. For a real parameter $t > 1$, before the first t-quasi-reduction, each step is t–decreasing. So, by Lemma 4, given a real parameter

$t > 1$, any input basis (b_1, b_2) will be t-quasi-reduced within at most $\log_t M$ iterations. Then the next Lemma leads to a bound for the total number of steps of Gauss' algorithm. Notice that the Lemma does not suppose that the input basis is integral. It is often used in the following sections.

Lemma 6. *Let $t \in]1, \sqrt{3}[$ be a real parameter. During the execution of Gauss' algorithm, there are at most two t–non-decreasing steps.*

Proof. At the first t–non-decreasing step, suppose that the test is negative. (A positive test means the end of the algorithm.) The basis is then (b_1, b_2) t-quasi-reduced and by Definition 2 and with the usual notation of (1),

$$(b_1, b_2) \text{ is proper } (|m_{21}| < 1/2) \quad \text{and} \quad |b_1|^2/t^2 \le |b_2|^2 < |b_1|^2. \tag{5}$$

After the swap, the matrix m and the orthogonalized basis b^* are updated:

Before the swap, $m = \begin{array}{c c} & \begin{array}{cc} b_1^* & b_2^* \end{array} \\ \begin{array}{c} b_1 \\ b_2 \end{array} & \begin{pmatrix} 1 & 0 \\ m_{21} & 1 \end{pmatrix} \end{array}$. After the swap, $\hat{m} = \begin{array}{c c} & \begin{array}{cc} \hat{b_1^*} & \hat{b_2^*} \end{array} \\ \begin{array}{c} b_2 \\ b_1 \end{array} & \begin{pmatrix} 1 & 0 \\ \hat{m}_{21} & 1 \end{pmatrix} \end{array}$,

with $\hat{m}_{21} = m_{21}(|b_1|^2/|b_2|^2)$. By relations (5), when $t \in]1, \sqrt{3}[$, $|\hat{m}_{21}| < t^2/2 < 3/2$, and $\lfloor \hat{m}_{21} \rceil \in \{-1, 0, 1\}$. So, the vector b_1 is replaced by $b_1 + \varepsilon b_2$, with $\varepsilon \in \{-1, 0, 1\}$. The new basis $(b_2, b_1 + \varepsilon b_2)$ is easily expressed in the old orthogonal basis (b_1^*, b_2^*) as follows

$$\begin{cases} |b_2|^2 = (m_{21})^2\, |b_1^*|^2 + |b_2^*|^2 \\ |b_1 + \varepsilon b_2|^2 = (1 + \varepsilon m_{21})^2\, |b_1^*|^2 + \varepsilon^2\, |b_2^*|^2 \end{cases},$$

and $\forall \varepsilon \in \{-1, 0, 1\}$,the new test will be positive: $|b_1 + \varepsilon b_2|^2 \ge |b_2|^2$.
(If $\varepsilon = 0$, it is true by relation (5). Else, $\varepsilon^2 = 1$ and since $|m_{21}| \le 1/2$, one gets $(1 + \varepsilon m_{21})^2 \ge m_{21}^2$.)

Hence, the number of iterations of Gauss' algorithm on an integer input basis of length $\log M$ is always less than $\log_{\sqrt{3}} M + 2$. This bound is not optimal [Val91]. However, in next sections we generalize this argumentation to the case of an arbitrary fixed dimension.

4 The 3–dimensional case

Let t be a real parameter such that $1 < t < 3/2$. Here, we count separately the iterations that are inside t–phases and the iterations that are not inside t–phases.
First, we show that the total number of steps that are not inside t–phases is *linear* with respect to the input length $\log_t M$ (Lemma 8). Second we prove that the total number of iterations inside t–phases is always less than nine (Lemma 9). Thus, we exhibit for the first time a linear bound for the number of iterations of the LLL algorithm in 3–dimensional space (the naive bound is M^6.) In addition, our argumentation gives a precise characterization of a reduced basis in the three–dimensional space.

Theorem 7. *The number of iterations of the LLL(1) algorithm on an input integer basis (b_1, b_2, b_3) of length M is less than*

$$\log_{\sqrt{3}} M + 6 \log_{3/2} M + 9.$$

Let us appreciate the previous upper–bound. The linear order for the bound is in fact optimal since it is so in dimension 2 [Val91] and one can obviously build from a basis of $n-1$ vectors b of maximal length M, another basis b' of n vectors of the same maximal length such the number of iterations of the LLL algorithm on the second basis is strictly greater than on the first basis. Moreover, even if we have not tried here to give the best coefficient of linearity in dimension 3, our bound is quite acceptable since [vS94] exhibits a family of bases of lengths $\log M$, for which the number of iterations of the algorithm is *greater* than $2.6 \log_2 M + 4$. Our bound is $13.2 \log_2 M + 9$. Observe that the classical bound on the usual non–optimal LLL($2/\sqrt{3}$) is $28.9 \log_2 M + 2$, and even computed more precisely as in Lemma 8 it remains $24.1 \log_2 M + 2$. So, our bound which is also valid for LLL(δ) with $\delta < 3/2$ improves the classical upper–bound on the number of steps of LLL(δ) provided that $\delta < 1.3$.

4.1 From an arbitrary basis to a quasi–reduced one

We begin with the following Lemma, which is a more precise version of Lemma 4 in the particular case of 3 dimensions.

Lemma 8. *Let the LLL algorithm run on a integer basis* (b_1, b_2, b_3) *of length* $\log M$. *Let t be a real parameter such that* $1 < t < \sqrt{3}$. *The number of steps that are not inside any t–phase is less than* $\log_{\sqrt{3}} M + 6 \log_t M$.

Proof. During an execution of the algorithm, let d_1 (respectively d_2) denote the determinant of the lattice generated by b_1 (respectively (b_1, b_2)). So d_1^2, d_2^2 are strictly positive integers and we have

$$1 < d_1 = |b_1^*| = |b_1| < M \quad \text{and} \quad 1 < d_2 = |b_1^*||b_2^*| < M^2. \tag{6}$$

First, notice that any step with a positive test modifies neither d_1 nor d_2. By hypothesis, all steps with negative tests and with the current index 2 ($i = 2$) are t–decreasing: They make d_2 decrease by at least by $1/t$ and from (6), we deduce that the total number of such steps is less than $2 \log_t M$. So is also the total number of phases that are not t-phases. Now consider the steps with current index 1 ($i = 1$). There is exactly one such step with a positive test per phase. On the other hand, thanks to Lemma 6, all the steps of current index 1 and with negative test are $\sqrt{3}$–decreasing, unless at most one per phase. Any $\sqrt{3}$–decreasing step of index 1 make d_1 decrease by at least by $1/t$. So, by (6), the total number of such steps is less than $\log_{\sqrt{3}} M$. This ends the proof of the Lemma.

In particular, for $1 < t \leq 3/2, \quad k \leq \log_{\sqrt{3}} M + 6 \log_{\frac{3}{2}} M$.

4.2 From a quasi–reduced basis to a reduced one

Lemma 9. *Let t be a real parameter such that* $1 < t < 3/2$. *When the dimension is fixed at* 3, *there are at most three t–phases during an execution of the algorithm. The total number of steps inside t–phases is less than nine.*

Proof. The proof is based on Lemma 10 and Corollaries 14 and 12. Lemma 10 shows that a t–phase of type I is necessarily an ending phase. Such a phase has exactly 3 iterations.
The central role in the proof is played by Lemma 13 and its Corollary 14 that show that when the dimension is fixed to 3, there are at most 2 t–phases of type II during an execution.
Finally, Lemma 11 and Corollary 12 show that any t–phase of type II has a maximum number of 3 iterations. These facts are used to make the proof clearer but they are not essential in the proof: actually, if a phase has more than 3 iterations, then these additional steps[4] are t–decreasing and all t–decreasing steps are already counted by Lemma 8.
Remarks. 1. If the parameter t is chosen closer to 1 ($\sqrt{6/5}$ rather than $3/2$), it can be rigorously shown (see [Akh99]) that if during the execution of the algorithm, a t-reduced basis is obtained, then a reduced one will be obtained after at most 9 steps. (A t–phase is necessarily followed by another t–phase) Such a requirement on t is not interesting, since it makes worse the final upper bound on the total number of iterations.
2. Some different argumentation will be done for the general case of fixed arbitrary dimension. Of course, this general argumentation holds here (when the dimension is fixed at 3), but the bound so obtained will be less precise.

Lemma 10. *Let t be a real parameter with* $1 < t < \sqrt{3}$. *A t-phase of type I has exactly 3 steps and is necessarily an ending phase.*

Proof. By hypothesis the vector b_1 will not be modified during such a phase. Since (b_1, b_2, b_3) is t–qr, so is (in particular) the basis $((b_2)_{\perp b_1}, (b_3)_{\perp b_1})$. By Lemma 6 (Gauss' algorithm), the basis $((b_3)_{\perp b_1}, (b_2)_{\perp b_1})$ will be reduced after only two iterations[5]. But here, there is one additional step between these two iterations (a step of current index 1 and with a positive test).

Lemma 11. *For any real parameter t such that* $t > 1$, *if a basis* $(b_1, b_2 \dots, b_{n-1}, b_n)$ *is t–quasi–reduced, then the basis* $(b_1, b_2 \dots, b_{n-1})$ *is* t'*–quasi–reduced, with* $t' > (2/\sqrt{3})t$.

Corollary 12. *Let t be a real parameter such that* $1 < t < 3/2$. *In* 3 *dimensions, a t–phase of type II has exactly* 3 *steps.*

Proof. The first test of this phase is obviously negative. By the previous Lemma, since (b_1, b_2, b_3) is $3/2$–quasi–reduced, then (b_1, b_3) is $\sqrt{3}$–quasi–reduced. Then by Lemma 6 (Gauss' algorithm), the basis (b_1, b_3) will be reduced after two iterations of test $(-, +)$.

The next Lemma plays a central role in the whole proof. This result which remains true when (b_1, b_2, b_3) is reduced gives also a precise characterization of a 1-Lovász reduced basis in dimension 3. A detailed proof is available in [Akh99].

[4] They are necessarily with negative tests and with the index i equal to 1.
[5] Lemma 6 does not demand the t–Gauss reduced basis to be integral.

Lemma 13. *Let t be a real parameter such that $1 < t < 3/2$. If the basis (b_1, b_2, b_3) is t–quasi–reduced and proper, then among all the vectors of the lattice that are not in the plan $\langle b_1, b_2\rangle$, there is at most one pair of vectors $\pm u$ whose lengths are strictly less than $|b_3|$.*

Proof. Let $u := x\, b_1 + y\, b_2 + z\, b_3$ be a vector of the lattice ($(x, y, z) \in \mathbb{Z}^3$). The vector u is expressed in the orthogonal basis b^* defined by (1) and its length satisfies $|u|^2 = (x + y\, m_{21} + z\, m_{31})^2\, |b_1^*|^2 + (y + z\, m_{32})^2\, |b_2^*|^2 + |b_3^*|^2$.
First, since (b_1, b_2, b_3) is $3/2$–quasi reduced, one gets easily that if $|z| > 1$ or $|y| > 1$ or $|x| > 1$, then $|u| > |b_3|$.
Now, if $z = 1$, by considering the ratio $|u|^2/|b_3|^2$, one can show (for a detailed proof, see [Akh99]) that there exits at most one pair $(x, y) \in \{0, 1, -1\}^2 \backslash \{(0, 0)\}$ such that $|u| < |b_3|$. This unique vector depend on the signs of m_{21}, m_{31} et m_{32}. Table 1 recapitulate the situation.

m_{21}	m_{31}	m_{32}	u
$+$	$+$	$+$	$b_3 - b_2$
$+$	$+$	$-$	$b_3 - b_1 + b_2$
$+$	$-$	$+$	$b_3 + b_1 - b_2$
$+$	$-$	$-$	$b_3 + b_2$
$-$	$+$	$+$	$b_3 - b_2 - b_1$
$-$	$+$	$-$	$b_3 + b_2$
$-$	$-$	$+$	$b_3 - b_2$
$-$	$-$	$-$	$b_3 + b_2 + b_1$

Table 1. The unique vector u candidate to be strictly shorter than b_3, as a function of signs of m_{21}, m_{31} and m_{32}.

Corollary 14. *Let t be a real parameter such that $1 < t < (3/2)$. During an execution of LLL(1) algorithm, there are at most 2 t–phases of type II.*

Proof. Assume (b_1, b_2, b_3) is the t–quasi–reduced basis at the beginning of a first t–phase of type II and and let (b_1, b_2, b_3') denote the basis obtained from (b_1, b_2, b_3) by making the latter proper.
Since the t-phase is of type II, $|b_3'| < |b_1|$ and the algorithm swaps $|b_1|$ and $|b_3'|$.
As (b_1, b_2) is Gauss-reduced, b_1 is a shortest vector of sub-lattice generated by (b_1, b_2). Thus $|b_3'| < |b_1|$ shows that there is no vector strictly shorter than b_3' in the sub-lattice generated by (b_1, b_2). On the other hand, the previous Lemma shows that there is at most one pair of vectors $\pm u$ of the lattice, outside the plan $\langle b_1, b_2\rangle$ whose lengths are strictly less than $|b_3'|$. Finally, in the whole lattice there is at most one pair of vectors $\pm u$ strictly shorter than b_3'. Thus, the vector b_3' can be swapped only once. In particular, only one new t'–phase (for any $t' > 1$) of type II may occur before the end of the algorithm.

Remark. For $1 < t < 3/2$, the previous Corollary shows that all phases that follow the first t–phase of type II, except eventually one, have exactly two iterations.

5 Arbitrary fixed dimension n

In the previous section, we argued in an *additive* way. Actually we chose a tractable value t_0 ($3/2$ in the 3–dimensional case) such that for $1 < t < t_0$ we could easily bound from above the total number of steps inside all t–phases. (This bound was 9 in the last Section). Then we *added* the last bound to the total number of iterations that were not inside t–phases.
Here, we will argue differently. On one hand, the total number of t–decreasing steps is classically upper bounded by $n^2 \log_t M$ (Lemma 4). Now, for a real parameter $t > 1$, let us call a *t–non–decreasing* sequence, a sequence of t–non–decreasing consecutive steps. During such a sequence just before any negative test of index i, the basis $(b_1, \ldots, b_{i+1})$ is t–quasi–reduced. The matter is that during a *t–non–decreasing* sequence, we cannot *quantify* efficiently the decreasing of the usual integer potential function[6] D (whose definition is recalled in (4)). The crucial point here (Lemma 17) is that for any real parameter $1 < t < 2$, there exists some integer $c(n, t)$ such that any t–non–decreasing sequence of the LLL(1) algorithm – when it works on an arbitrary input basis $(b_1, \ldots, b_n)$ (no matter the lengths of the vectors)– has strictly less than $c(n, t)$ steps. In other words any sequence of iterations, which is longer than $c(n, t)$ has a t–decreasing step.
Hence, our argumentation is somehow *multiplicative* since the total number of iterations with negative tests is thus bounded from above by $c(n, t)n^2 \log_t M$. We deduce the following theorem which for the first time exhibits a linear bound on the number of iterations of the LLL algorithm in fixed dimension.

Theorem 15. *For any fixed dimension n, the maximum number of iterations of the optimal LLL algorithm is linear with respect to the the length of the input. More precisely, (i) for all constant $A > (2/\sqrt{3})^{(1/6)}$, the number of iterations of the algorithm when it runs on an integer input basis $(b_1, \ldots, b_n)$ whose length of vectors is less than M is O($A^{n^3} \log M$). Moreover, (ii) under a very plausible heuristic, it is also O($(2/\sqrt{3})^{n^2/2} \log M$).*

The first formulation (i) is based on Proposition 17, and Lemmata 4, 16. The proof of the second formulation (ii) uses also Lemma 18 (which is proved under a very plausible heuristic).
The next Lemma is an adaptation of ones used by Babai, Kannan and Schnorr [Bab86,Kan83,Sch87] when finding a shortest vector in a lattice with a Lovász–reduced basis on hand.

Lemma 16. *Let $t \in]1, 2[$ be a real parameter and $\mathcal{L}$ be a lattice generated by a basis $b := (b_1, \ldots, b_n)$, which is not necessarily integral and whose vectors are of arbitrary*

[6] The naive bound is obtained using only the fact that D^2 is a strictly positive integer less than $M^{n(n-1)}$ and it is strictly decreasing at each step with a negative test.

length. If b is proper and t–quasi-reduced then there exists an integer $\alpha(n,t)$ such that the number of vectors of the lattice $\mathcal{L}$ whose lengths are strictly less than $|b_1|$ is strictly less than $\alpha(n,t)$. Moreover,

$$\alpha(n,t) < \sqrt{\frac{3t^2}{4-t^2}}\, 3^{n-1} \left(\frac{2}{\sqrt{3}}\right)^{\frac{n(n-1)}{2}} . \tag{7}$$

Proof. Any vector $v = \sum_{i=1}^{n} v_i b_i$ is expressed in the orthogonal basis (b^*) as shown by relation (1). If the vector v is shorter than b_1, it satisfies:

$$|v|^2 = \sum_{i=1}^{n} \left(\sum_{j=i}^{n} v_j m_{j,i} \right)^2 |b_i^*|^2 < |b_1|^2.$$

To find a vector shorter than b_1, we have to find an integer vector $(v_1, \ldots, v_n)$ satisfying $0 \leq |v_n| < |b_1|/|b_n^*|$ and

$$\left(\sum_{j=i}^{n} v_j m_{j,i} \right)^2 |b_i^*|^2 < |b_1|^2 - \sum_{k=i+1}^{n} \left(\sum_{j=k}^{n} v_j m_{j,i} \right)^2 |b_k^*|^2, \quad \text{for } i = n-1, \ldots, 1.$$

As $|\sum_{j=i}^{n} v_j m_{j,i}| < |b_1|/|b_i^*|$, the number of possible values for v_i when $v_{i+1}, \ldots, v_n$ are fixed is at most $\lfloor 2|b_1|/|b_i^*| \rfloor + 1$. Thus, at most

$$\prod_{i=2}^{n} \left(\lfloor 2\frac{|b_1|}{|b_i^*|} \rfloor + 1 \right)$$

vectors $(v_1, \ldots, v_n)$ may be shorter than b_1. Now, recall that $(b_1, \ldots, b_n)$ is t–quasi–reduced. Then, Definition (2) involves

$$\frac{|b_i^*|}{|b_{i+1}^*|} \leq \frac{2}{\sqrt{3}}, \quad \text{for } i = 1, \ldots, n-2 \text{ and } \frac{|b_{n-1}^*|}{|b_n^*|} \leq \frac{2t}{\sqrt{4-t^2}}.$$

The last relations lead to the bound for $\alpha(n,t)$ exhibited by relation (7).

Remark. The sequence $\alpha(n,t)$ is increasing with n.

Proposition 17. *Let n be a fixed dimension and t a real parameter in $]1, \sqrt{3}[$. There exists an integer $c(n,t)$ such that the length of any t–non–decreasing sequence of the LLL(1) algorithm – on any input basis $(b_1, \ldots, b_n)$, no matter the lengths of its vectors and no matter the basis is integral – is strictly less than $c(n,t)$.*

Proof. We prove the Lemma by induction on n. The case $n = 2$ is trivial and $c(2,t) = 2$ (Lemma 6). Suppose that the assertion holds for any basis of $n-1$ vectors and let the algorithm run on a basis $b := (b_1, \ldots, b_n)$. After at most $c(n-1,t)$ steps, if the algorithm has not finished, there is either a t-decreasing step, or a t–non–decreasing

step with the current index i satisfying $i = n - 1$. [7] In the second case, at this moment, the basis $(b_1, \ldots, b_n)$ is t–quasi–reduced.
We first show that if the next phase if of type I, then the t–non–decreasing sequence will be finished after at most $c(n-1,t) + \alpha(n-1,t)$ more steps. In fact, in this case the algorithm works actually in a $(n-1)$–dimensional lattice generated by the basis

$$b_{\perp b_1} := ((b_2)_{\perp b_1}, \ldots, (b_n)_{\perp b_1}) \tag{8}$$

which is also t–quasi–reduced. By the induction hypothesis, while the algorithm works on $b_{\perp b_1}$ the length of a t–non–decreasing sequence is less than $c(n-1,t)$. During these $c(n-1,t)$ steps, each change of the first vector $(b_2)_{\perp b_1}$ (no more than $\alpha(n-1,t)$, by Lemma 16) is followed by one step (of current index one) with a positive test which has not been counted yet.
Now suppose the next phase is of type II. By Lemma 16, the first vector of the t–quasi–reduced basis $(b_1, \ldots, b_n)$ can be modified at most $\alpha(n,t)$ times. So, there are at most $\alpha(n,t)$ successive phases of type II and each of them has no more than $c(n-1,t)$ steps, since after the first negative test, the algorithm works actually on $(b_1, \ldots, b_{n-1})$.
Finally , after the last t–phase of type II, it may be one more t-phase of type I, whose number of steps is less than $c(n-1,t) + \alpha(n-1,t)$. From all this, and by recalling that $\alpha(n,t)$ is increasing with respect to n, we have

$$c(n,t) \le c(n-1,t) + c(n-1,t)\alpha(n,t) + c(n-1,t) + \alpha(n-1,t) < (c(n-1,t)+1)(\alpha(n,t)+2),$$

$$\text{and finally} \quad c(n,t) + 1 \le (c(2,t)+1) \prod_{i=2}^{n} (\alpha(i,t)+2)\,. \tag{9}$$

Proof (First formulation (i) of Theorem 15). Each sequence of $c(n,t)$ steps contains at least one t–decreasing step. At each t–decreasing step, the quantity D, which is always in the interval $[1, M^{n^2}]$, is decreasing by at least $1/t$. So the total number of iterations of the algorithm is always less than $c(n,t)n^2 \log_t M$. Now by choosing a fixed $t \in]1, \sqrt{3}]$, relation (7) together with relation (9) show that $n^2 c(n,t)$ is bounded from above by $A_1^{n^3}$ where A_1 an absolute constant slightly greater than $(2/\sqrt{3})^{1/6}$.

In the first proof, we choose for t an arbitrary value in the interval $]1,2[$. Now, we improve our bound by choosing t as a function of n. What we really need here is to evaluate the number of possible successive t-phases of type II. So the main question is: When a basis $(b_1, \ldots, b_n)$ of a lattice L is t–quasi–reduced, how many lattice points u are satisfying $(1/t)|b_1| < |u| < |b_1|$? More precisely, is it possible to choose t, as a function of dimension n, such that the open volume between the two n–dimensional balls of radii $|b_1|$ and $(1/t)|b_1|$ does not contain any lattice point?
Now, we answer these questions under a quite reasonable heuristic principle which is often satisfied. So, the bound on $C(n)$ and on the maximum number of iterations will be improved. This heuristic is due to Gauss. Consider a lattice of determinant $\det(L)$. The

[7] Else, there would be a t–non–decreasing of more than $c(n-1,t)$ steps while the algorithm runs on the basis $(b_1, \ldots, b_{n-1})$.

heuristic claims that the number of lattice points inside a ball $\mathcal{B}$ is well approximated by volume$(\mathcal{B})/\det(L)$. More precisely the error is of the order of the surface of the ball $\mathcal{B}$. This principle holds for very large class of lattices, in particular those used in applications (for instance "regular lattices" where the minima are close to each other [VGT88] and where the fundamental parallelepiped is close to a hyper–cube). Moreover, notice that this heuristic leads also to the result of Lemma 16.
Under this assumption and if $\gamma_n = \pi^{n/2}/\Gamma(1+n/2)$ denotes the volume of the n–dimensional unit ball, then the number of lattice points $\beta(n,t)$ that lie strictly between balls of radii $|b_1|$ and $(1/t)|b_1|$ satisfies (at least asymptotically)

$$\beta(n,t_n) \leq \gamma_n \frac{|b_1|^n}{\det(L)}(1-\frac{1}{t^n}). \tag{10}$$

Since $(b_1,\ldots,b_n)$ is t–quasi–reduced and $\det L = \prod_{i=1}^{n} |b_i^*|$, then for all $t < \sqrt{3}$,

$$\frac{|b_1|^n}{det(L)} = \frac{|b_1|^n}{|b_1^*||b_2^*|\ldots|b_n^*|} \leq 3^n \left(\frac{2}{\sqrt{3}}\right)^{\frac{n(n-1)}{2}}.$$

Then, using the classical Stirling approximation, $\beta(n,t)$ is bounded from above :

$$\beta(n,t) < 3\left(\frac{\pi e}{n}\right)^{n/2}\left(\frac{2}{\sqrt{3}}\right)^{\frac{n(n-1)}{2}}(1-\frac{1}{t^n}).$$

By routine computation, we deduce the following Lemma.

Lemma 18. *Suppose that there exists n_0 such that for $n > n_0$, relation (10) is true. Then if t_n is defined by*

$$\frac{1}{t_n} := 1 - \frac{1}{3n}\left(\frac{n}{\pi e}\right)^{n/2}\left(\frac{\sqrt{3}}{2}\right)^{\frac{n(n-1)}{2}}, \tag{11}$$

for all $n > n_0$, we have

$$\beta(n,t_n) < 1 \quad \textit{and} \quad \frac{1}{\log t_n} < 3n\left(\frac{\pi e}{n}\right)^{n/2}\left(\frac{2}{\sqrt{3}}\right)^{\frac{n(n-1)}{2}}.$$

Remark: The sequence t_n defined by (11) is decreasing and tends to 1 with n. So $\beta(n-1,t_n) = \beta(n-1,t_{n-1}) = 0$.

Proof (Second formulation (ii) of Theorem 15). The quantities t_n and n_0 are defined by the previous Lemma. First we prove that for $n > n_0$ and with the notations of Lemma 17, we have

$$c(n,t_n) \leq c(n-1,t_n) + (c(n_0,t_n) + \alpha((n_0,t_n)). \tag{12}$$

Indeed, after $c(n-1,t)$ steps, the basis $(b_1,\ldots,b_n)$ is t_n–quasi–reduced. Now if H denotes the vector space $\langle b_1,\ldots,b_{n-n_0-2}\rangle$ the (n_0+1)–dimensional basis

$$((b_{n-n_0-1})_{\perp H},\ldots,(b_n)_{\perp H}) \tag{13}$$

is t_n–quasi–reduced as well. Thus, by the previous Lemma, during the t_n–non–decreasing sequence, its first vector cannot be modified. So, from the first time that the basis $(b_1, \ldots, b_n)$ is t_n–quasi–reduced until the end of the t_n–non–decreasing sequence the current index i will always be in the integral interval $\{n - n_0, \ldots, n\}$. Since, the algorithm works actually in a n_0–dimensional basis defined by (13), the sequence of t–non–decreasing iterations may continue for at most $c(n_0, t_n) + \alpha(n_0, t_n)$ more iterations.[8] This ends the proof of relation (12). So, for $n > n_0$,

$$\text{So, for } n > n_0, \quad c(n, t_n) \leq (n - n_0 + 1)(c(n_0, t_n) + \alpha(n_0, t_n)).$$

Since $t_n < t_{n_0}$, the basis considered in (13) is also t_{n_0}–quasi–reduced and by Lemma 16, $\alpha(n_0, t_n) \leq \alpha(n_0, t_{n_0})$. (The same relation is true for $n < n_0$.) Finally the quantity $c(n_0, t_n) + \alpha(n_0, t_n)$ is a constant B that depends only on n_0. We have then $c(n, t_n) < nB$. Hence, if a sequence is longer than nB then it contains a t_n–decreasing step and the total number of iterations of the algorithm is less than $nBn^2 \log_{t_n} M$. Finally Lemma 18 gives an upper–bound for $1/\log t_n$ and leads to the formulation (ii) of Theorem 15.

6 Conclusion

Our paper gives for the first time linear bounds for the the maximum number of iterations of the optimal LLL algorithm, in fixed dimension.
I believe that the complexity of finding an optimal Lovász–reduced basis is of great interest and not well–known.
Kannan presented [Kan83] an algorithm which uses as sub–routine the non–optimal LLL algorithm ($\delta > 1$) and outputs a Korkine–Zolotarev basis of the lattice in $O(n^n) \log M$ steps. Such an output is also an optimal Lovász–reduced basis (Actually it is stronger). Thus, Kannan's algorithm provides an upper–bound on the complexity of finding an optimal Lovász–reduced basis[9]. For the future, one of the two following possibilities (or both) has to be considered.
(1) Our upper–bound is likely to be improved. However, observe that in this paper we have already improved notably the naive bound for fixed dimension (the exponential order is replaced by linear order). For the moment our bound remains worse than the one Kannan exhibits for his algorithm,
(2) The LLL algorithm which is the most natural way to find an optimal Lovász–reduced basis is not the best way (and then the same phenomenon may be possible for finding a non–optimal Lovász–reduced basis: more efficient algorithms than the classical LLL algorithm may output the same reduced basis).

Acknowledgments. I am indebted to Brigitte Vallée for drawing my attention to algorithmic problems in lattice theory and for regular helpful discussions. I wish to thank her also for her help to improve this paper.

[8] The quantity $\alpha(n_0, t_n)$ corresponds to the maximum number of positive tests with index $i = n - n_0$ after the the first t_n–quasi–reduction.

[9] Moreover, so far as I know, there exists no polynomial time algorithm for finding a Korkine–Zolotarev reduced basis from an an optimal Lovász–reduced basis. So finding an optimal Lovász–reduced basis seems to be *strictly* easier than finding a Korkine–Zolotarev reduced basis.

References

[Ajt97] M. Ajtai. The shortest vector problem in L_2 is NP-hard for randomized reduction. *Electronic Colloquim on Computationnal Complexity*, 1997. available via: http://www.eccc.uni-trier.de/eccc/.

[Akh99] A. Akhavi. Complexité de l'algorithme LLL pour la valeur optimale du paramètre d'approximation. Technical report, Rapprort GREYC, Caen, 1999.

[Bab86] L. Babai. On Lovász' lattice reduction and the nearest lattice point problem. *Combinatorica*, 6(1):1–13, 1986.

[BK84] A. Bachem and R. Kannan. Lattices and the basis reduction algorithm. *CMU–CS*, pages 84–112, 1984.

[Cai99] J. Cai. Some recent progress on the complexity of lattice problems. Technical report, ECCC, 1999.

[DFV97] H. Daudé, Ph. Flajolet, and B. Vallée. An average–case analysis of the Gaussian algorithm for lattice reduction. *Combinatorics, Probability and Computing*, 123:397–433, 1997.

[Kan83] R. Kannan. Improved algorithm for integer programming and related lattice problems. In *15th Ann. ACM Symp. on Theory of Computing*, pages 193–206, 1983.

[Kan87] R. Kannan. Algorithmic geometry of numbers. *Ann. Rev. Comput. Sci.*, 2:231–267, 1987.

[KS96] M. Kaib and C. P. Schnorr. The generalized Gauss reduction algorithm. *J. of Algorithms*, 21:565–578, 1996.

[Lag80] J. C. Lagarias. Worst–case complexity bounds for algorithms in the theory of integral quadratic forms. *J. Algorithms*, 1:142–186, 1980.

[Lag83] J. C. Lagarias. Solving low-density subset problems. *IEEE*, 1983.

[Len83] H.W. Lenstra. Integer programming with a fixed number of variables. *Math. Oper. Res.*, 8:538–548, 1983.

[LLL82] A. K. Lenstra, H. W. Lenstra, and L. Lovász. Factoring polynomials with rational coefficients. *Math. Ann.*, 261:513–534, 1982.

[LO83] J. C. Lagarias and A. M. Odlyzko. Solving low–density subset sum problems. In *24th IEEE Symposium FOCS*, pages 1–10, 1983.

[Mic98] D. Micciancio. The Shortest Vector in a lattice is hard to approximate to within some constant. In *to appear in Proc. 39-th Symposium On Foundations of Computer Science*, 1998.

[Sch87] C. P. Schnorr. A hierarchy of polynomial time lattice basis reduction algorithm. *Theoretical Computer Science*, 53:201–224, 1987.

[Sch95] C. P. Schnorr. Attacking the Chor-Rivest cryptosystem by improved lattice reduction. In *Eurocrypt*, 1995.

[Val89] B. Vallée. Un problème central en géométrie algorithmique des nombres: la réduction des réseaux. Autour de l'algoritme LLL. *Informatique Théorique et Applications*, 3:345–376, 1989. English translation by E.Kranakis, CWI Quaterly 1990-3. Amsterdam.

[Val91] B. Vallée. Gauss' algorithm revisited. *Journal of Algorithms*, 12:556–572, 1991.

[vEB81] P. van Emde Boas. Another NP-complete problem and the complexity of finding short vectors in a lattice. *Rep. 81-04 Math. Inst. Univ. Amsterdam*, 1981.

[VGT88] B. Vallée, M. Girault, and Ph. Toffin. How to braek Okamoto's cryptosystem by reducing lattice bases. In *Proceedings of Eurocrypt*, 1988.

[vS94] O. von Sprang. *Basisreduktionsalgorithmen fü Gitter kleiner Dimension*. PhD thesis, Universität des Saarlandes, 1994.