HAL
open science

Another view of the Gaussian algorithm

Ali Akhavi, Céline Moreira

To cite this version:

Ali Akhavi, Céline Moreira. Another view of the Gaussian algorithm. Latin American Informatics'04, Apr 2004, Buenos Aires, Argentina. pp.474–487. hal-00159666

HAL Id: hal-00159666
https://hal.science/hal-00159666v1

Submitted on 4 Jul 2007

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

A REWRITE BASED ANALYSIS OF ALGORITHMS

ALI AKHAVI AND CÉLINE MOREIRA DOS SANTOS

Abstract. We introduce here a new method for extracting worst–cases of algorithms by using rewrite systems over automorphisms groups of inputs.
We propose a canonical description of an algorithm, that is also related to the problem it solves. The description identifies an algorithm with a set of a rewrite systems over the automorphisms groups of inputs. All possible execution of the algorithm will then be reduced words of these rewriting system.
Our main example is reducing two-dimensional Euclidean lattice bases. We deal with the Gaussian algorithm that finds shortest vectors in a two–dimensional lattice. We introduce four rewrite systems in the group of unimodular matrices, *i.e.* matrices with integer entries and with determinant equal to ± 1 and deduce a new worst-case analysis of the algorithm that generalizes Vallée's result[16] to the case of the usual Gaussian algorithm. An interesting (but not easy) future application will be lattice reduction in higher dimensions, in order to exhibit a tight upper- bound for the number of iterations of LLL–like reduction algorithms in the worst case.
Sorting ordered finite sets are here as a nice esay example to illustrate the purpose of our method. We propose several rewrite systems in the group $\mathcal{S}$ of permutations and canonically identify a sorting algorithm with a rewrite system over $\mathcal{S}$. This brings us to exhibit worst-cases of several sorting algorithms.

hal-00159666, version 1 - 4 Jul 2007

1. Introduction

A Euclidean lattice is the set of all integer linear combinations of a set of linearly independent vectors in $\mathbb{R}^p$. The independent vectors are called *a basis* of the lattice. Any lattice can be generated by many bases. All of them have the same cardinality, that is called *the dimension* of the lattice. If B and B' represent matrices of two bases of the same lattice in the canonical basis of $\mathbb{R}^p$, then there is a unimodular matrix U such that $B' = UB$. A unimodular matrix is a matrix with integer entries and with determinant equal to ± 1.
The lattice basis reduction problem is to find bases with good Euclidean properties, that is, with sufficiently short vectors and almost orthogonal.

In two dimensions, the problem is solved by the Gaussian algorithm, that finds in any two–dimensional lattice, a basis formed with the shortest possible vectors. The worst–case complexity of Gauss' algorithm (explained originally in the vocabulary of quadratic forms) was first studied by Lagarias [7], who showed that the algorithm is polynomial with respect to its input. The worst–case complexity of Gauss' algorithm was also studied later more precisely by Vallée[16].

In 1982, Lenstra, Lenstra and Lovász [11] gave a powerful approximation reduction algorithm for lattices of arbitrary dimension. Their famous algorithm, called LLL, was an important breakthrough to numerous theoretical and practical problems in computational number theory and cryptography: factoring polynomials with rational coefficients [11], finding linear Diophantine approximations [8], breaking various cryptosystems [4] and integer linear programming [6, 9]. The LLL algorithm is a possible generalization of its 2–dimensional version, which is the Gaussian algorithm.
The LLL algorithm seems difficult to analyze precisely, both in the worst–case[1, 10, 11] and in average–case[2, 3]. In particular when the dimension is higher than two, the problem of the real worst–case of the algorithm is completely open. However, LLL–like reduction algorithms are so widely used in practice that the analyzes are a real challenge, both from a theoretical and practical point of view.
The purpose of our paper is a new approach to the worst- -case analyze of LLL–like lattice reduction algorithms. For the moment this approach is presented only in two dimensions. We have to observe here that the worst case of some variant of the Gaussian algorithm is already known [16]. Even if our paper generalize this knowledge to the case of the usual Gaussian algorithm, we do not consider it as the most important point of this paper. Our aim here is to present this new approach.

An LLL–like lattice reduction algorithm uses some (finite) elementary transforms. We consider the group generated by these basic transforms. Then we exhibit a family of rewriting rules over this group, corresponding to the mechanism of the algorithm. The rewriting rules make some forbidden sequences and the length of a valid word over the set of generators becomes very close to the number of steps of the algorithm. This makes appear the smallest length of input demanding a given number of iterations to the reduction algorithm.
From a combinatorial point of view, the group of n– dimensional lattice transformations $GL_n(\mathbb{Z})$, *i.e.* the multiplicative group of $n \times n$ matrices with determinant ± 1, is the group of automorphisms of the free Abelian group on n free generators[1]. Here we are concerned by $GL_2(\mathbb{Z})$, which is is well–known and whose presentation in terms of generators and relators is known since the nineteenth century.
In this paper we present a rewriting system over $GL_2(\mathbb{Z})$, that makes us *predict* how the Gaussian algorithm is running on an arbitrary input. We deduce from this the worst–case configuration of the usual Gaussian algorithm and give an "optimal" maximum for the number of steps of the Gaussian algorithm. Our result generalizes the result of Vallée [16]. She studied a variant of the Gaussian algorithm where elementary transforms made by the algorithm are some integer matrices of determinant equal to 1. In the case of the usual Gaussian algorithm, elementary transforms are integer matrices of determinant either 1 or -1.
Il the following we briefly outline the two steps of our method.

1.1. **First step.** Consider a deterministic algorithm A that run on an input x (the data x is a set X of data). Then by mean of elementary transforms taken in a set $F \subset X^X$, the algorithm changes the input step by step ($x \to f(x)$ until the modified data satisfies some output condition ($x \in O \subset X$. An elementary or atomic transform is a transform that cannot be decomposed by the algorithm:

$$\forall f \in F, \quad \forall k \in \mathbb{N}, k > 1, \quad \forall (f_1, \ldots, f_k) \in (F \backslash \{id\})^k, \quad f \neq \prod_{i=1}^{k} f_i \tag{1.1}$$

This is of course a very general context containing both iterative and recursive algorithms.
Algorithm A :
  Input: $x \in X$.
  Output: $y \in O$.
Initialization: i $:=$ **1**;
While $x \notin O$ **do**
  Determine an adequate function $f \in F$ by a computation on x and eventually on i.
  x:=f(x)
  i:=i+1
The algorithm A is deterministic. We suppose that the determination of the adequate function $f \in F$ at a moment i (which may depend on the history of the execution) has a cost that can be added without ambiguity to the cost of the function f . Considering F as an alphabet, the set F^* of finite words on F is then the monoid generated by the set of *free* generators F. F^* contain all (finite) executions of the algorithm.
  Now fix a sequence of transforms $(f_1, f_2, \ldots, f_k) \in F^k$.

> *Is the sequence* $(f_1, f_2, \ldots, f_k) \in F^k$ *a possible execution for the algorithm?* More precisely, is there $(x, y) \in X \times Y$ such that the algorithm A outputs y when running on an input x and following the exact sequence of transforms $(f_1, f_2, \ldots, f_k)$?

Answering this question in such a general context is very difficult and the general problem (formulated more precisely) is likely undecidable. However the answer in restricted class of algorithms bring indeed a strong understanding of the mechanism of the algorithm and we believe that it is interesting by its own with lots of possible applications in program verifying, or program designing.
In this paper, we propose a method to answer this question in the case of Gaussian algorthm and three sorting algorithms. A set (finite in the case of our examples) of rewriting systems encode all possible executions of a given algorithm. All possible executions will be the normal forms (or reduced forms) of these rewriting systems.

[1]By the free Abelian group, we mean that the only non trivial relators (*i.e.* the additional relators compared to the free group) are the commutators.

1.2. **second step.** Usually when counting the number of steps of an algorithm, one considers all inputs of length less than a fixed bound, say M. Then one estimates the maximum number of steps taken over all these inputs by:

$$f(M) := \max_{\text{all inputs of length less than } M} \text{number of steps of the algorithm.} \tag{1.2}$$

[2]

Here to exhibit the precise real worst–case, we first proceed in "the opposite way". Consider k a fixed number of steps. We will estimate the minimum length of those inputs demanding at least k steps to be processed by the algorithm:

$$g(k) := \min_{\text{all inputs demanding at least } k \text{ steps}} \text{length of the input.} \tag{1.3}$$

Clearly $f(g(k)) = k$. Otherwise there would be an input of length less than $g(k)$ demanding more than k steps. But $g(k)$ is by definition the minimal length of such inputs. So by inverting the fuction g , we can compute f.

2. Gaussian algorithm and the new approach to its worst–case analysis

Endow $\mathbb{R}^2$ with the usual scalar product $(\ ,\ )$ and Euclidean length $|\mathbf{u}| = (\mathbf{u}, \mathbf{u})^{1/2}$. A two–dimensional lattice is a discrete additive subgroup of $\mathbb{R}^2$. Equivalently, it is the set of all integer linear combinations of two linearly independent vectors. Generally it is given by one of its bases $(\mathbf{b}_1, \mathbf{b}_2)$. Let $(\mathbf{e}_1, \mathbf{e}_2)$ be the canonical basis of $\mathbb{R}^2$. We often associate to a lattice basis $(\mathbf{b}_1, \mathbf{b}_2)$ a matrix B, such that *the vectors of the basis are the rows of the matrix*:

$$B = \begin{array}{c} \\ \mathbf{b}_1 \\ \mathbf{b}_2 \end{array} \begin{array}{c} \begin{array}{cc} \mathbf{e}_1 & \mathbf{e}_2 \end{array} \\ \begin{pmatrix} b_{1,1} & b_{1,2} \\ b_{2,1} & b_{2,2} \end{pmatrix} \end{array}. \tag{2.1}$$

The *length L of the previous basis* (or the *length of the matrix B*) is defined here to be the maximum of $(|\mathbf{b}_1|, |\mathbf{b}_2|)$.
The usual Gram–Schmidt orthogonalization process builds, in polynomial–time, from a basis $b = (\mathbf{b}_1, \mathbf{b}_2)$ an orthogonal basis $b^* = (\mathbf{b}_1^*, \mathbf{b}_2^*)$ and a lower–triangular matrix M that expresses the system b into the system b^*[3]. Put $m = \frac{(\mathbf{b}_2, \mathbf{b}_1)}{(\mathbf{b}_1, \mathbf{b}_1)}$. By construction, the following equalities hold:

$$\left\{ \begin{array}{lcl} \mathbf{b}_1^* & = & \mathbf{b}_1 \\ \mathbf{b}_2^* & = & \mathbf{b}_2 - m\,\mathbf{b}_1 \end{array} \right. , \ M = \begin{array}{c} \\ \mathbf{b}_1 \\ \mathbf{b}_2 \end{array} \begin{array}{c} \begin{array}{cc} \mathbf{b}_1^* & \mathbf{b}_2^* \end{array} \\ \begin{pmatrix} 1 & 0 \\ m & 1 \end{pmatrix} \end{array}. \tag{2.2}$$

The ordered basis $B = (\mathbf{b}_1, \mathbf{b}_2)$ is called *proper* if the quantity m satisfies

$$-1/2 \leq m < 1/2. \tag{2.3}$$

There is a natural and unique representative of all the bases of given two–dimensional lattice. This basis is composed of two shortest vectors generating the whole lattice. It is called the Gauss– reduced basis and the Gaussian algorithm outputs this reduced basis running on any basis of the lattice. Any lattice basis in two dimensions can always be expressed as

$$B = U\,R, \tag{2.4}$$

where R is the so–called Gaussian reduced basis of the same lattice and U is a unimodular matrix, *i.e.* an element of $GL_2(\mathbb{Z})$. The goal of a reduction algorithm, the Gaussian algorithm in two dimensions, is to find R given B. The Gaussian algorithm is using two kinds of elementary transforms, explained in the sequel of this paper. Let (b_1, b_2) be an input basis of a lattice and the matrix B expressing (b_1, b_2) in the canonical basis of $\mathbb{R}^2$ as specified by (2.1).
The algorithm first makes an integer translation of b_2 in the direction of b_1 in order to make b_2 as short as possible. This is done just by computing the integer x nearest to $m = (b_2, b_1)/(b_1, b_1)$ and replacing b_2 by $b_2 - xb_1$. Notice that, after this integer translation, the basis (b_1, b_2) is proper.

[2]When dealing with a non–trivial algorithm f is always an increasing function.
[3]Of course, b^* is generally not a basis for the lattice generated by b.

The second elementary transform is just the swap of the vectors b_1 and b_2 in case when after the integer translation we have $|b_1| > |b_2|$.
The algorithm iterates these transforms, until after the translation, b_1 remains still smaller than b_2, *i.e.*, $|b_1| \leq |b_2|$.
The Gaussian algorithm can also be regarded (especially for the analysis purposes) as an algorithm that gives a decomposition of the unimodular matrix U of relation (2.4) by means of some basic transforms:

$$\begin{array}{ll} \text{Input:} & B = U\,R. \\ \text{Output:} & R = T^{x_{k+1}} S T^{x_k} S T^{x_{k-1}} \dots S T^{x_2} S T^{x_1} B; \end{array} \tag{2.5}$$

where

$$S = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \quad \text{and} \quad T = \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix}. \tag{2.6}$$

The matrix T corresponds to an integer translation of b_2 in the direction of b_1 by one. Of course, we have:

$$T^x = \prod_{i=1}^{x} \text{ and } T = \begin{pmatrix} 1 & 0 \\ x & 1 \end{pmatrix}, \text{ for all } x \in \mathbb{Z}.$$

The matrix S represents a swap. *Each step of the algorithm is indeed an integer translation followed by a swap*[4]. So each step of the Gaussian algorithm is represented by ST^x, $x \in \mathbb{Z}^*$.
Writing the output in this way (2.5) shows not only the output but how precisely the algorithm is working since T and S represent the only elementary transforms made during the execution of the Gaussian algorithm. So when studying the mechanism of a reduction algorithm in two dimensions and for a fixed reduced basis R, the algorithm can be regarded as a decomposition algorithm over $GL_2(\mathbb{Z})$:

$$\begin{array}{ll} \text{Input:} & U \in GL_2(\mathbb{Z}). \\ \text{Output:} & \text{a decomposition of } U,\ U := T^{x_{k+1}}\, ST^{x_k}\, ST^{x_{k-1}} \dots ST^{x_2}\, ST^{x_1}. \end{array} \tag{2.7}$$

The integer k denotes the number of steps. Indeed the algorithm terminates[1, 7, 16]. In the sequel we will prove that the above mechanism does not depend strongly on the reduced basis R. More precisely there are exactly 4 rewrite systems. (for all reduced bases of $\mathbb{R}^2$)

The unimodular group in two dimensions $GL_2(\mathbb{Z})$ has been already studied [12, 13, 14, 15] and it is well–known that $\{S, T\}$ is a possible family of generators for $GL_2(\mathbb{Z})$. Of course there are relators associated to these generators and there is no unicity of the decomposition of an element of $GL_2(\mathbb{Z})$ in terms of S and T. But the Gaussian algorithm gives one precise of these possible decompositions. In the sequel of this paper, we will completely characterize this decomposition and we will call it *the Gaussian decomposition of a unimodular matrix*. Roughly speaking, we exhibit forbidden sequences of values for the x_i–s.
More precisely, we exhibit in Section 3 a set of rewriting rules that lead to the formulation output by the Gaussian algorithm, from any product of matrices involving S and T. The precise characterization of the Gaussian decomposition that we give makes appear the slowest manner the length of a unimodular matrix can grow with respect to its Gaussian decomposition. More precisely we consider unimodular matrices the length of Gaussian decomposition of which is fixed, say k:

$$U := T^{x_{k+1}}\, ST^{x_k}\, ST^{x_{k-1}} \dots ST^{x_2}\, ST^{x_1}.$$

We exhibit in Section 4 the Gaussian word of length k with minimal length. We naturally deduce the minimum length $g(k)$ of all inputs demanding at least k steps (Section 5). Finally by "inverting" the function g we find the maximum number of steps of the Gaussian algorithm.

3. The Gaussian decomposition of a unimodular matrix

Let Σ be a (finite or infinite) set. A word ω on Σ is a finite sequence

$$\alpha_1 \alpha_2 \dots \alpha_n \tag{3.1}$$

where n is a positive integer, and $\alpha_i \in \Sigma$, for all $i \in \{1, \dots, n\}$. Let Σ^* be the set of finite words on Σ. We introduce for convenience the *empty word* and we denote it by 1.

[4] A priori x_1 and x_{k+1} in (2.5) may be zero so the algorithm may start by a swap or finish by a translation.

Consider the alphabet $\Sigma = \{S, T, T^{-1}\}$. Remember that we call Gaussian decomposition, the expression of U output by the Gaussian algorithm. Remember also that, for a given basis B, there exists a unique couple (U, R) such that U and R are output by the Gaussian algorithm while reducing B.

Lemma 1. *Let $R = (b_1, b_2)$ be a reduced basis. Then one of the following cases occurs:*

- $|b_1| < |b_2|$ *and* $m \neq -1/2$*;*
- $|b_1| = |b_2|$ *and* $m \neq -1/2$*;*
- $|b_1| < |b_2|$ *and* $m = -1/2$*;*
- $|b_1| = |b_2|$ *and* $m = -1/2$*.*

Now consider a word ω on Σ, that is, an element of Σ^*, a unimodular matrix U and a reduced basis R. We give, in the following subsections, sets of rewriting rules depending on the form of R, such that any word in which none of these rewriting rules can be applied is Gaussian. Since the results of these subsections are very similar, we only give detailed proofs for Subsection 3.1 in the appendix.

3.1. **The basis R is such that $|b_1| < |b_2|$ and $m \neq -1/2$.** Say that ω is a *reduced word* or a *reduced decomposition of the unimodular matrix U*, if ω is a decomposition of U in which no one of the rewriting rules of Theorems 1 can be applied.
Thus, Theorem 1 shows that, for a given reduced basis R such that $|b_1| < |b_2|$ and $m \neq -1/2$, the Gaussian decomposition and a reduced decomposition of a unimodular matrix are the same, which implies that this decomposition is unique.

Theorem 1. *Let ω_1 be any decomposition of U in terms of the family of generators $\{S, T\}$. The Gaussian decomposition of U is obtained from ω_1 by applying repeatedly the following set of rules:*

$$S^2 \longrightarrow 1; \tag{3.2}$$

$$T^x T^y \longrightarrow T^{x+y}; \tag{3.3}$$

$$\forall x \in \mathbb{Z}_-^*, \qquad ST^2ST^x \longrightarrow TST^{-2}ST^{x+1}; \tag{3.4}$$

$$\forall x \in \mathbb{Z}_+^*, \qquad ST^{-2}ST^x \longrightarrow T^{-1}ST^2ST^{x-1}; \tag{3.5}$$

$$\forall x \in \mathbb{Z}^*, \forall k \in \mathbb{Z}_+, \ ST\,ST^x \prod_{i=k}^{1} ST^{y_i} \longrightarrow TST^{-x-1} \prod_{i=k}^{1} ST^{-y_i}; \tag{3.6}$$

$$\forall x \in \mathbb{Z}^*, \forall k \in \mathbb{Z}_+, \ ST^{-1}ST^x \prod_{i=k}^{1} ST^{y_i} \longrightarrow T^{-1}ST^{-x+1} \prod_{i=k}^{1} ST^{-y_i}. \tag{3.7}$$

The trivial rules (3.2) and (3.3) have to be applied whenever possible. So any word ω_1 on the alphabet Σ can trivially be written as

$$T^{x_{k+1}} \prod_{i=k}^{1} ST^{x_i}, \tag{3.8}$$

with $x_i \in \mathbb{Z}^*$ for $2 \leq i \leq k$ and $(x_1, x_{k+1}) \in \mathbb{Z}^2$. The integer k is called *the length*[5] *of* ω_1. Notice that usually the length of a word as in (3.1) is n, which would corresponds here to $2k + 1$. Here the length is k, which correspond to the number of iterations of the algorithm minus 1.

The proof of Theorem 1 is given in appendix. It consists in the following lemmas:

- Lemma 2, where we prove that the rewriting process terminates;
- Lemma 3, where we prove that any reduced word is also Gaussian;

[5]The length of a word has of course to be distinguished with what we call the length of a unimodular matrix, that is the maximum of absolute values of its coefficients.

- Lemmas 4 and 5, where we prove that the use of a nontrivial rewriting rules changes a base of the lattice in another base of the same lattice.

Proofs of Theorems 2, 3 and 4, which are given in the following subsections, are very similar.

3.2. The basis R is such that $|b_1| = |b_2|$ and $m \neq -1/2$.

Theorem 2. *Let ω_1 be any decomposition of U in terms of the family of generators $\{S, T\}$. The Gaussian decomposition of U is obtained from ω_1 by applying repeatedly the set of rules* (3.2) *to* (3.7) *of Theorem 1, together with the following rules:*

$$\forall x \in \mathbb{Z}^*, \forall k \in \mathbb{Z}_+,\ ST\,ST^x \left(\prod_{i=k}^{1} ST^{y_i}\right) \longrightarrow T\,ST^{-x-1} \left(\prod_{i=k}^{1} ST^{-y_i}\right) T; \tag{3.9}$$

$$\forall x \in \mathbb{Z}^*, \forall k \in \mathbb{Z}_+,\ ST^{-1}ST^x \left(\prod_{i=k}^{1} ST^{y_i}\right) \longrightarrow T^{-1}ST^{-x+1} \left(\prod_{i=k}^{1} ST^{-y_i}\right) T; \tag{3.10}$$

$$\text{if } \omega_1 = \omega\, S,\ \text{then } \omega S \longrightarrow \omega; \tag{3.11}$$

$$\text{if } \omega_1 = \omega\, ST,\ \text{then } \omega ST \longrightarrow \omega T\,ST^{-1}. \tag{3.12}$$

3.3. The basis R is such that $|b_1| < |b_2|$ and $m = -1/2$.

Theorem 3. *Let R be a reduced basis and let U be a unimodular matrix,* i.e., *an element of $GL_2(\mathbb{Z})$. Let ω_1 be any decomposition of U in terms of the family of generators $\{S, T\}$. The Gaussian decomposition of U is obtained from ω_1 by applying repeatedly the rules* (3.2) *to* (3.5) *of Theorem 1 until ω_1 is reduced in the sense of Theorem 1. Then, if we have $\omega_1 = \omega\, ST^2\, S$, the following rule applies:*

$$\omega ST^2\, S \longrightarrow \omega T\, ST^{-2}\, ST, \tag{3.13}$$

and the rewriting process is over.

3.4. The basis R is such that $|b_1| = |b_2|$ and $m = -1/2$.

Theorem 4. *Let R be a reduced basis and let U be a unimodular matrix,* i.e., *an element of $GL_2(\mathbb{Z})$. Let ω_1 be any decomposition of U in terms of the family of generators $\{S, T\}$. The Gaussian decomposition of U is obtained from ω_1 by applying repeatedly Rules* (3.2) *to* (3.5) *of Theorem 1, together with Rules* (3.9) *and* (3.10) *and the following set of rules:*

$$\text{if } \omega_1 = \omega\, S,\ \text{then } \omega S \longrightarrow \omega; \tag{3.14}$$

$$\text{if } \omega_1 = \omega\, ST,\ \text{then } \omega ST \longrightarrow \omega T; \tag{3.15}$$

$$\text{if } \omega_1 = \omega\, ST^2,\ \text{then } \omega ST^2 \longrightarrow \omega T\,ST^{-1}. \tag{3.16}$$

4. The length of a unimodular matrix with respect to its Gaussian decomposition

Let $B = (b_1, b_2)$ be a basis. The *length* of B, denoted by $\ell(B)$, is the sum of the squares of the norms of its vectors, that is, $\ell(B) = |b_1|^2 + |b_2|^2$.

The easy but tedious proof of the following theorem is given in the appendix, see Lemmas 6, 7, 8, 9 and 10.

Theorem 5. *Let $R = (b_1, b_2)$ be a reduced basis, let k be a positive integer, and let $x_1, \ldots, x_{k+1}$ be integers such that the word $\omega = T^{x_{k+1}} \prod_{i=k}^{1} ST^{x_i}$ is Gaussian. Then the following properties hold:*

1. *if $|b_1| < |b_2|$ and $m \geq 0$ then $\ell(\omega R) \geq \ell((ST^{-2})^{k-1}\, S\, R)$;*
2. *if $|b_1| < |b_2|$ and $-1/2 < m < 0$ then $\ell(\omega R) \geq \ell((ST^{2})^{k-1}\, S\, R)$;*
3. *if $|b_1| < |b_2|$ and $m = -1/2$ then $\ell(\omega R) \geq \ell((ST^{-2})^{k-1}\, ST\, R)$;*
4. *if $|b_1| = |b_2|$ then $\ell(\omega R) \geq \ell((ST^{-2})^{k-1}\, ST^{-1}\, R)$.*

5. The maximum number of steps of the Gaussian algorithm

Theorem 6. *Let $k > 2$ be a fixed integer. There exists an absolute constant A such that input basis demanding more than k steps to the Gaussian algorithm has a length greater than $A(1+\sqrt{2})^k$:*

$$g(k) \geq A(1+\sqrt{2})^k.$$

It follows that any input with length less than $A(1+\sqrt{2})^k$ is demanding less than k steps. We deduce the following corollary.

Corollary 1. *There is an absolute constant A such that the number of steps of the Gaussian algorithm on inputs of length less than M is bounded from above by*

$$\log_{(1+\sqrt{2})}\left(\frac{M}{A}\right).$$

6. Sorting algorithms

Let n be a positive integer, and let $[1,\dots,n]$ be the sorted list of the n first positive integers. Let $\mathcal{S}_n$ be the set of all permutations on $[1,\dots,n]$, and let $\mathcal{S}$ be the set of all permutations on a list of distinct integers of variable size. Let us denote by t_i the transposition which swaps the elements in positions i and $i+1$ in the list , for all $i \in \{1,\dots,n\}$. Any permutation can be written in terms of the t_i-s. Put $\Sigma_n = \{t_1,\dots,t_n\}$ and $\Sigma = \{t_i \colon i \in \mathbb{N}^*\}$. Thus Σ_n (resp. Σ) is a generating set of $\mathcal{S}_n$ (resp. $\mathcal{S}$).

As in previous sections, any word ω on Σ will be denoted as following:

$$\omega = t_{i_1} t_{i_2} \dots t_{i_k} = \prod_{j=1}^{k} t_{i_j},$$

where k and i_1, …, i_k are positive integers.

Definition 1. Let $\omega_1 = t_{i_1} t_{i_2} \dots t_{i_k}$ and $\omega_2 = t_{j_1} t_{j_2} \dots t_{j_l}$ be words on Σ.

(1) The *length of* ω, denoted by $|\omega|$, is k;
(2) the *distance between* ω_1 *and* ω_2, denoted by $\mathrm{Dist}(\omega_1,\omega_2)$, is given by $\min_{t_i\in\omega_1, t_j\in\omega_2} |i-j|$;
(3) the *maximum* (resp. *minimum*) of ω_1, denoted by $\max(\omega_1)$ (resp. $\min(\omega_1)$), is given by $\max_{t_i\in\omega_1}(i)$ (resp. $\min_{t_i\in\omega_1}(i)$);
(4) ω_1 is an *increasing word* (resp. *decreasing word*) whether $i_p < i_{p+1}$ (resp. $i_p > i_{p+1}$), for all $p \in \{1,\dots,k-1\}$;
(5) ω_1 is a *consecutively increasing word* (resp. *consecutively decreasing word*) whether $i_{j+1} - i_j = 1$ (resp. $i_j - i_{j+1} = 1$), for all $j \in \{1,\dots,k-1\}$;
(6) $\omega_1 < \omega_2$ (resp. $\omega_1 > \omega_2$) whether ω_1 and ω_2 are increasing (resp. decreasing) words such that $\max(\omega_1) \leq \min(\omega_2)$ (resp. $\min(\omega_1) \geq \max(\omega_2)$);
(7) ω_1 is *minimal on the left in* $\omega_2\omega_1$ (resp. *maximal on the right in* $\omega_1\omega_2$) whether ω_1 is an increasing word such that $i_1 \leq j_l$ (resp. $j_1 \leq i_k$);
(8) similarly, ω_1 is *minimal on the right in* $\omega_2\omega_1$ (resp. *maximal on the left in* $\omega_1\omega_2$) whether ω_1 is a decreasing word such that $j_l \leq i_1$ (resp. $i_k \leq j_1$).

It is easy to prove that any word ω on Σ can be uniquely written on the form

$$\omega = \omega_1\omega_2\dots\omega_m, \tag{6.1}$$

where ω_i is an increasing (resp. decreasing) word maximal on the right and on the left, for all $i \in \{1,\dots,m\}$. We will call (6.1) the *increasing decomposition* (resp. *decreasing decomposition*) of ω, and we will denote it by $[\omega_1,\dots,\omega_m]$. We define $s\colon \Sigma^* \to \mathbb{N}$ as the map given by the rule

$$s(\omega) = m,$$

where $[\omega_1,\dots,\omega_m]$ is the increasing decomposition of ω. Moreover, it is also easy to prove that ω_i can be uniquely written on the form

$$\omega_i = \omega'_{i_1}\omega'_{i_2}\dots\omega'_{i_{p_i}},$$

where the ω'_j-s are consecutively increasing (resp. decreasing) and minimal on the left (resp. right), for all $i \in \{1,\dots,m\}$. The decomposition $[\omega'_{i_1},\omega'_{i_2},\dots,\omega'_{i_{p_i}}]$ is called the *consecutively increasing decomposition* (resp. *consecutively decreasing decomposition*) of ω_i.

6.1. **Bubble sort.** The basic idea of the bubble sort algorithm is the following: pairs of adjacent values in the list to be sorted are compared and interchanged if they are out of order, the process starting from the beginning of the list. Thus, list entries 'bubble upward' in the list until they bump into one with a higher sort value.

The algorithm first compares the two first elements of the list and swap them if they are in the wrong order. Then, the algorithm compares the second and the third elements of the list and swaps them if necessary. The algorithms continues to compare adjacent elements from the beginning to the end of the list. This whole process is iterated until no changes are done.

Let σ be a permutation on $[1,\dots,n]$. There exists a unique decomposition ω of σ on the alphabet Σ corresponding to the sequence of elementary transforms performed by the bubble sort algorithm on $\sigma[1,\dots,n]$. We will call it the *bubblian decomposition* of σ. Notice that $(\omega)^{-1}\sigma = 1$.

$$\begin{array}{ll} \text{Input:} & \sigma \in \mathcal{S}. \\ \text{Output:} & \text{a decomposition of } \sigma,\ \sigma := t_1 \dots t_m. \end{array} \tag{6.2}$$

Definition 2. A word ω on Σ is a *bubblian word* if it corresponds to a possible execution of the bubble sort algorithm.

Let us define some rewriting rules on Σ^*. In the following equations, i, j and k are arbitrary positive integers and ω is a word on Σ:

$$t_i\, t_i \longrightarrow 1; \tag{6.3}$$

$$\text{if } \mathrm{Dist}(i+1,\omega) > 1, \text{ then } t_{i+1}\,\omega\, t_i\, t_{i+1} \longrightarrow \omega t_i\, t_{i+1}\, t_i; \tag{6.4}$$

$$\text{if } \mathrm{Dist}(i+1,\omega) > 1 \text{ and } \omega \text{ is maximally increasing, then } \omega\, t_i \longrightarrow t_i\, \omega; \tag{6.5}$$

$$\text{if } \mathrm{Dist}(j,k\omega) > 1 \text{ and either } i \le j \le k \text{ or } k < i \le j, \text{ then } t_i\, t_k\, \omega\, t_j \longrightarrow t_i\, t_j\, t_k\, \omega. \tag{6.6}$$

Theorem 7. *Let σ be a permutation and let $\omega \in \Sigma^*$ be a decomposition of σ on Σ. The bubblian decomposition of σ is obtained from ω by applying repeatedly the rules* (6.3) *to* (6.6).

Remark 1. Let ω and ω' be words on Σ. It is well known that a presentation of $\mathcal{S}$ on Σ is the following:

- $t_i\, t_i = 1$;
- $t_i\, t_j = t_j\, t_i$;
- $t_i\, t_{i+1}\, t_i = t_{i+1}\, t_i\ t_{i+1}$;

for all positive integers i, j such that $|i-j| = 1$. Thus, it is easy to prove that if ω' is obtained from ω, then $\omega = \omega'$ in $\mathcal{S}$.

The sketch of the proof of Theorem 7 is very similar to the proof of Theorem 1 and is given in the appendix. Notice that we can easily deduce from Theorem T:redbubble the worst-case for the bubble sort algorithm.

6.2. **Other iterative sorting algorithms.** We also give without proof some rewriting rules for the insertion sort algorithm and the selection sort algorithm, see the appendix.

7. Conclusion

In this paper we studied the Gaussian algorithm by considering a rewriting system over $GL_2(\mathbb{Z})$. We first believe that our method should be applied to other variants of the Gaussian algorithm (for example, Gaussian algorithm with other norms [5]) and for each variant there is an adequate rewriting system over $GL_2(\mathbb{Z})$. The most important and interesting continuation to this work is to generalize the approach in higher dimensions. Even in three dimensions the worst–case configuration of all possible generalization of the Gaussian algorithm is completely unknown for the moment. ([17] has tried without success in 3 dimensions.) Although the problem is really difficult, we have already achieved a step, since the LLL algorithm uses the Gaussian algorithm as an elementary transform.

The group of n–dimensional lattice transformations has been studied first by Nielsen [14] ($n = 3$) and for an arbitrary n by Magnus [12, 13], based on the work of Nielsen[15]. Their work should certainly help to exhibit such rewriting systems on $GL_n(\mathbb{Z})$ if there exists.
This approach may also be an insight to the still open problem of the complexity of the optimal LLL algorithm[1, 10].

Acknowledgments. The authors are indebted to Brigitte Vallée for drawing their attention to algorithmic problems in lattice theory and for regular helpful discussions.

References

[1] A. Akhavi. Worst–case complexity of the optimal LLL algorithm. In *Proceedings of LATIN'2000 - Punta del Este*. LNCS 1776, pp 476–490.
[2] H. Daudé, Ph. Flajolet, and B. Vallée. An average-case analysis of the Gaussian algorithm for lattice reduction. *Comb., Prob. & Comp.*, 123:397–433, 1997.
[3] Daudé, H., and Vallée, B. An upper bound on the average number of iterations of the LLL algorithm. *Theoretical Computer Science 123(1)* (1994), pp. 95–115.
[4] A. Joux and J. Stern. Lattice reduction: A toolbox for the cryptanalyst. *J. of Cryptology*, 11:161–185, 1998.
[5] M. Kaib and C. P. Schnorr. The generalized Gauss reduction algorithm. *J. of Algorithms*, 21:565–578, 1996.
[6] R. Kannan. Improved algorithm for integer programming and related lattice problems. In *15th Ann. ACM Symp. on Theory of Computing*, pages 193–206, 1983.
[7] J. C. Lagarias. Worst-case complexity bounds for algorithms in the theory of integral quadratic forms. *J. Algorithms*, 1:142–186, 1980.
[8] J. C. Lagarias. The computational complexity of simultaneous D iophantine approximation problems *SIAM J. Computing*, 14:196–209, 1985.
[9] H.W. Lenstra. Integer programming with a fixed number of variables. *Math. Oper. Res.*, 8:538–548, 1983.
[10] H.W. Lenstra. Flags and lattice basis reduction. In *Procedings of the 3rd European Congress of Mathematics - Barcelona July 2000* I: 37-51, Birkhäuser Verlag, Basel
[11] A. K. Lenstra, H. W. Lenstra, and L. Lovász. Factoring polynomials with rational coefficients. *Math. Ann.*, 261:513–534, 1982.
[12] W. Magnus. Über n-dimensionale G ittertransformationen. *Acta Math.*, 64:353–357, 1934.
[13] W. Magnus, A. Karrass, and D. Solitar. Combinatorial group theory. Dover, New York, 1976 (second revised edition).
[14] J. Nielsen. Die Gruppe der dreidimensionale G ittertransformationen. *Kgl Danske Videnskabernes Selskab., Math. Fys. Meddelelser*, V 12: 1–29, 1924.
[15] J. Nielsen. Die Isomorphismengruppe der freien Gruppen. *Math. Ann.*, 91:169–209, 1924. translated in english by J. Stillwell in *J. Nielsen collected papers*, Vol 1.
[16] B. Vallée. Gauss' algorithm revisited. *J. of Algorithms*, 12:556–572, 1991.
[17] O. von Sprang. *Basisreduktionsalgorithmen für Gitter kleiner Dimension*. PhD thesis, Universität des Saarlandes, 1994.

Appendix A. The Gaussian decomposition of a unimodular matrix

In the following proofs, we will essentially use the set of rules (3.4), (3.5), (3.6), (3.7) and apply (3.2) and (3.3) implicitly whenever possible.
So with any initial ω_1, we always obtain a reduced word after applying a finite number of times the rewriting rules. Moreover, no matter in which order the different rewriting rules are used, the same unique reduced word – corresponding to a Gaussian word – is always obtained from ω_1, as proved by the following lemmas.

Lemma 2. *Let ω_1 be a word as in* (3.8)*. Then the rewriting process always terminates*[6]*.*

The proof of Lemma 2 will use the following notations.

Notation 1. Let k be a nonnegative integer, and let $x_1, \ldots, x_{k+1}$ be integers such that $x_2, \ldots, x_k$ are nonzero. Put $\omega_1 = T^{x_{k+1}} \prod_{i=1}^{k} ST^{x_i}$. We denote by ω_1^- the word $T^{-x_{k+1}} \prod_{i=1}^{k} ST^{-x_i}$. Put

$$S_1 = \{i \colon 2 \le i \le k \text{ and } |x_i| = 1\};$$
$$S_2 = \{i \colon 2 \le i \le k,\ x_i x_{i-1} < 0 \text{ and } |x_i| = 2\}.$$

We also put $d(\omega_1) = \sum_{i \in S_1 \cup S_2} i$.

Proof. We proceed by induction on the length of ω_1, and on the sum $d = d(\omega_1)$. The property is trivially true whether $|\omega_1| \in \{0, 1, 2\}$ and $d \in \mathbb{N}$.

Let k be a positive integer such that $k \ge 2$. Suppose that the property holds for any word of length k. Suppose that $|\omega_1| = k + 1$. The property holds whether d belongs to $\{0, 1\}$. Suppose that the property holds

[6]Of course saying that the rewriting process presented by the previous Theorem always terminates has a priori nothing to do with the well-known fact that the Gaussian algorithm always terminates.

for any word ω of length $k+1$ such that $d > d(\omega) \geq 1$. Let i be in $S_1 \cup S_2$. If $x_i = 1$ (resp. $x_i = -1$), then we use the rule (3.6) (resp. (3.7)), and the length of ω_1 strictly decreases. Suppose now that $x_i = 2$. Then ω_1 can be written as $\omega_2\, ST^x\, ST^2\, ST^y\, \omega_3$, where $x \in \mathbb{Z}$, $y \in \mathbb{Z}_-^*$, ω_2 and ω_3 are words on the alphabet Σ, such that ω_2 does not end by a S and ω_3 is either 1 or starts by S. If we use the rule (3.4), then we get the word $\omega_1' = \omega_2\, ST^{x+1}\, ST^{-2}\, ST^{y+1}\, \omega_3$. Put $d' = d(\omega_1')$, and $\delta = \sum_{j \in S_1 \cup S_2 \setminus \{i+1, i, i-1\}} j$. Notice that if $x = 0$, then $\omega_1 = T^z\, ST^2\, ST^y\, \omega_3$, with $z \in \mathbb{Z}$.

Case 1. Suppose that either $x = -1$ or $y = -1$.

Then $|\omega_1'| = k$, and the rewriting process terminates.

Case 2. Suppose that $y \notin \{-3, -2, -1\}$.

- Suppose that $x \notin \{-2, -1, 1\}$: then $d' = d - 1$.
- Suppose that $x = -2$. Then the following equalities hold:

$$\omega_1 = \omega_2\, ST^{-2}\, ST^2\, ST^y\, \omega_3;$$
$$\omega_1' = \omega_2\, ST^{-1}\, ST^{-2}\, ST^{y+1}\, \omega_3.$$

  Thus, we have $d' = i + 1 < d = 2i + 1 + \delta$.
- Suppose that $x = 1$. Then the following equalities hold:

$$\omega_1 = \omega_2\, ST^1\, ST^2\, ST^y\, \omega_3;$$
$$\omega_1' = \omega_2\, ST^2\, ST^{-2}\, ST^{y+1}\, \omega_3.$$

  Thus we have $d' = i + 1 < d = 2i + 1 + \delta$.

Case 3. Suppose that $y = -3$.

- Suppose that $x \notin \{-2, -1, 1\}$. Then $d' \leq i - 1 + \delta < i + \delta \leq d$.
- Suppose that $x = -2$. Then

$$\omega_1 = \omega_2\, ST^{-2}\, ST^2\, ST^{-2}\, \omega_3;$$
$$\omega_1' = \omega_2\, ST^{-1}\, ST^{-2}\, ST^{-2}\, \omega_3.$$

  Thus $d' \leq 2i - 1 + \delta < 2i + 1 + \delta \leq d$.
- Similarly, if $x = 1$, then the following equalities hold:

$$\omega_1 = \omega_2\, ST^1\, ST^2\, ST^{-3}\, \omega_3$$
$$\omega_1' = \omega_2\, ST^2\, ST^{-2}\, ST^{-2}\, \omega_3.$$

  Thus $d' \leq 2i - 1 + \delta < 2i + 1 + \delta \leq d$.

Case 4. Suppose that $y = -2$.

- Suppose that $x \notin \{-2, -1, 1\}$. Then $d' \leq i - 1 + \delta < i + \delta \leq d$.
- Suppose that $x = -2$. Then the following equalities hold:

$$\omega_1 = \omega_2\, ST^{-2}\, ST^2\, ST^{-2}\, \omega_3;$$
$$\omega_1' = \omega_2\, ST^{-1}\, ST^{-2}\, ST^{-1}\, \omega_3.$$

  Thus $d' \leq 2i - 1 + \delta < 2i + 1 + \delta \leq d$.
- Suppose that $x = 1$. Then the following equalities hold:

$$\omega_1 = \omega_2\, ST^1\, ST^2\, ST^{-2}\, \omega_3$$
$$\omega_1' = \omega_2\, ST^2\, ST^{-2}\, ST^{-1}\, \omega_3.$$

  Thus $d' \leq 2i - 1 + \delta < 2i + 1 + \delta \leq d$.

By induction hypothesis, the rewriting process always terminates. □

Lemma 3. *Let B be the matrix of a proper basis (b_1, b_2) (see (2.1), (2.2) and (2.3)). Let $x \in \mathbb{Z}^*$ be a non zero integer and $\tilde{B} := ST^x B$ express the matrix of the basis $(\tilde{b}_1, \tilde{b}_2)$,* i.e. *$\tilde{b}_1$ is the first row of $\tilde{B}$ and $\tilde{b}_2$ is its second row.*

(1) *If $|x| \geq 3$, then $\tilde{B}$ is still proper. Moreover,*
   - *if (b_1, b_2) and x are both positive or both negative, then $\tilde{B}$ is proper whenever $|x| \geq 2$.*
   - *if B is reduced, $|b_1| < |b_2|$ and $m \neq -1/2$, then $\tilde{B}$ is proper for all $x \in \mathbb{Z}$.*

(2) *If $|x| \geq 2$, then $|\tilde{b}_2| < |\tilde{b}_1|$. Moreover, if (b_1, b_2) and x are both positive or both negative, it is true provided that $|x| \geq 1$.*

(3) *If $|x| \geq 2$, then $\max(|\tilde{b}_1|, |\tilde{b}_2|) \geq \max(|b_1|, |b_2|)$.*

(4) *If $|x| \geq 1$, then $(\tilde{b}_1, \tilde{b}_2)$ and x are both positive or both negative.*

Proof. The definitions of B, b^* and m are given by Relations (2.1), (2.2) and (2.3). We have $\tilde{b}_1 = b_2 + xb_1$ and $\tilde{b}_2 = b_1$. In the sequel we express all the needed quantities in the orthogonal basis b^*.

(1) We will show that $\left|\frac{(\tilde{b}_1, \tilde{b}_1)}{(\tilde{b}_1, \tilde{b}_2)}\right| \leq 2$.
Notice that
$$\frac{(\tilde{b}_1, \tilde{b}_1)}{|(\tilde{b}_1, \tilde{b}_2)|} = \frac{(m+x)^2|b_1^*|^2 + |b_2^*|^2}{|m+x||b_1^*|^2} < \frac{1}{|m+x|}.$$
Since $||x| - |m|| < |m+x|$ and $|m| \leq 1/2$, when $|x| \geq 3$, clearly $2 < |m+x|$ and when $mx > 0$ this is still true whenever $|x| \geq 2$.

(2) We have to show that $|b_1| < |b_2 + xb_1|$. We notice that
$$\frac{|b_1|^2}{|b_2 + xb_1|^2} = \frac{|b_1^*|^2}{(m+x)^2|b_1^*|^2 + |b_2^*|^2} < \frac{1}{(m+x)^2}.$$
If $mx < 0$, we have $1 < 1.5 \leq ||x| - |m|| < |m+x|$. If $mx > 0$, $|m+x| \geq 1$, $\forall x \geq 1$.

(3) From the last point, we know that $|b_1| \leq |b_2 + xb_1|$. If $|b_2| < |b_1|$, we have also $|b_2| < |b_2 + xb_1|$. (This is true for $|x| \geq 2$ or for $|x| \geq 1$ provided that $xm > 0$.
Now if $|b_2| \geq |b_1|$, since (b_1, b_2) is proper, it is indeed Gauss–reduced. So b_1 and b_2 are the shortest vector generating the whole lattice. So for all $x \neq 0$, $|b_2 + xb_1| > b_2$.

(4) By definition $(\tilde{b}_1, \tilde{b}_2) := (b_2 + xb_1, b_1) = (x+m)|b_1^*|^2$. Since $|m| \leq 1/2$ if $|x| \geq 1$, the quantities $x+m$ and x have the same sign.

□

Corollary 2. *Let ω_1 be a word of the form (3.8). Then there is a unique reduced word ω_1' corresponding to ω_1. Moreover, ω_1' is a Gaussian word.*

Lemma 4. *Let B be a proper basis. Put*
$$\tilde{B} := ST^2\, ST^x\, B \text{ and } B' := (T\, ST^{-2}\, ST^{x+1})^{-1}\tilde{B};$$
$$(\text{resp. } \tilde{B} := ST^{-2}\, ST^x\, B \text{ and } B' := (T^{-1}\, ST^2\, ST^{x-1})^{-1}\tilde{B}).$$
Then B' is such that $B' = -B$.

Proof. The following equalities hold:
$$ST^2\, ST^x = \begin{pmatrix} 2x+1 & 2 \\ x & 1 \end{pmatrix} \text{ and } T\, ST^{-2}\, ST^{x+1} = \begin{pmatrix} -2x-1 & -2 \\ -x & -1 \end{pmatrix},$$
which concludes the proof. □

Lemma 5. *Let R be a reduced basis such that $m \neq -\frac{1}{2}$. Put $\tilde{B} := ST\, ST^x\, \omega\, B$ (resp. $\tilde{B} := ST^{-1}\, ST^x\, \omega_1\, B$), where ω is a word on Σ of the form (3.8). Then the basis $B' := (T\, ST^{-x-1}\, \omega^-)^{-1}\tilde{B}$ (resp. $B' := (T^{-1}\, ST^{-x+1}\, \omega^-)^{-1}\tilde{B}$) is a reduced basis.*

Proof. Notice that:
$$ST\, ST^x = \begin{pmatrix} x+1 & 1 \\ x & 1 \end{pmatrix} \text{ and } T\, ST^{-x-1} = \begin{pmatrix} -x-1 & 1 \\ -x & 1 \end{pmatrix}.$$
Thus, it is easy to prove that $ST\, ST^x = T\, ST^{-x-1} \begin{pmatrix} -1 & 0 \\ 0 & 1 \end{pmatrix}$.

The proof of the following easy claim is left to the reader.

Claim 1. *Let y be an integer. Then the following equalities hold:*

$$\begin{pmatrix} -1 & 0 \\ 0 & 1 \end{pmatrix} ST^y = ST^{-y} \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \text{ and } \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} ST^y = ST^{-y} \begin{pmatrix} -1 & 0 \\ 0 & 1 \end{pmatrix}.$$

Thus, it is easy to prove by induction on the length of ω_1 that there exists an integer $\beta \in \{-1, 1\}$ such that $ST\, ST^x \omega_1 = T\, ST^{-x-1}\, \omega_1^-\, \beta \begin{pmatrix} -1 & 0 \\ 0 & 1 \end{pmatrix}$. Thus, we have $B' = \beta \begin{pmatrix} -1 & 0 \\ 0 & 1 \end{pmatrix} B$, which concludes the proof. □

Corollary 3. *Let $\omega_1 = T^{x_{k+1}} \prod_{i=k}^{1} ST^{x_i}$ be a non–reduced word, and let R be a reduced basis. If $\omega_1\, B$ is the input of the Gaussian algorithm, then the sequence of elementary transforms made during the execution is exactly represented by the reduced word uniquely associated to ω_1.*

Appendix B. The length of a unimodular matrix with respect to its Gaussian decomposition

The easy proof of the following lemma is left to the reader.

Lemma 6. *Let k be a positive integer. There exist nonnegative integers α, β and γ such that:*

$$(ST^2)^k = \begin{pmatrix} \alpha & \beta \\ \beta & \gamma \end{pmatrix}, (ST^{-2})^k = (-1)^k \begin{pmatrix} \alpha & -\beta \\ -\beta & \gamma \end{pmatrix} \text{ and } \alpha = 2\beta + \gamma.$$

Lemma 7. *Let B be a proper basis, and let x be an integer such that $|x| \geq 3$. Then $\ell(ST^x\, B) \geq \ell(ST^2\, B)$ and $\ell(ST^x\, B) \geq \ell(ST^{-2}\, B)$.*

Proof. Put $\tilde{B}_x = (\tilde{b}_{1,x}, \tilde{b}_{2,x}) = ST^x\, B$, for all integer x. Then

- $\tilde{b}_{1,x} = (x+m)b_1^* + b_2^*$;
- $\tilde{b}_{2,x} = b_1^*$.

It is obvious that $|\tilde{b}_{2,x}|^2 = |\tilde{b}_{2,2}|^2 = |\tilde{b}_{2,-2}|^2$. Moreover, we have

$$|\tilde{b}_{1,x}|^2 - |\tilde{b}_{1,2}|^2 = [(x+m)^2 - (2+m)^2]|b_1^*|^2 = [(x+2+2m)(x-2)]|b_1^*|^2 \geq 0,$$

and

$$|\tilde{b}_{1,x}|^2 - |\tilde{b}_{1,-2}|^2 = [(x+m)^2 - (m-2)^2]|b_1^*|^2 = [(x-2+2m)(x+2)]|b_1^*|^2 \geq 0,$$

which concludes the proof. □

Lemma 8. *Let B be a proper basis, let k be a positive integer, let $\varepsilon \in \{1, -1\}$ be an integer, and let x be an integer such that $|x| \geq 3$. The following properties hold:*

- *if x is positive, then $\ell((ST^2)^k\, ST^x\, B) \geq \ell((ST^{\varepsilon 2})^{k+1}\, B)$;*
- *if x is negative, then $\ell((ST^{-2})^k\, ST^x\, B) \geq \ell((ST^{\varepsilon 2})^{k+1}\, B)$.*

Proof. Put $(ST^2)^k = \begin{pmatrix} \alpha & \beta \\ \beta & \gamma \end{pmatrix}$ and $(ST^{-2})^k = (-1)^k \begin{pmatrix} \alpha & -\beta \\ -\beta & \gamma \end{pmatrix}$, as in Lemma 6. Put $\tilde{B} = (\tilde{b}_{1,x}, \tilde{b}_{2,x}) = (ST^2)^k\, ST^x\, B$ and $\tilde{B}' = (\tilde{b'}_{1,x}, \tilde{b'}_{2,x})$. Then

- $\tilde{b}_{1,x} = (\alpha(x+m) + \beta)b_1^* + \alpha b_2^*$, $\quad \tilde{b'}_{1,x} = (\alpha(x+m) - \beta)b_1^* + \alpha b_2^*$;
- $\tilde{b}_{2,x} = (\beta(x+m) + \gamma)b_1^* + \beta b_2^*$, $\quad \tilde{b'}_{2,x} = (-\beta(x+m) + \gamma)b_1^* - \beta b_2^*$;

Suppose that x is positive. Then

$$\begin{aligned} |\tilde{b}_{1,x}|^2 - |\tilde{b}_{1,2}|^2 &= [(\alpha(x+m) + \beta)^2 - (\alpha(2+m) + \beta)^2]|b_1^*|^2 \\ &= [\alpha(x-2)(\alpha(x+2+2m) + 2\beta)]|b_1^*|^2 \geq 0 \end{aligned}$$

and

$$\begin{aligned} |\tilde{b}_{2,x}|^2 - |\tilde{b}_{2,2}|^2 &= [(\beta(x+m) + \gamma)^2 - (\beta(2+m) + \gamma)^2]|b_1^*|^2 \\ &= [\beta(x-2)(\beta(x+2+2m) + 2\gamma)]|b_1^*|^2 \geq 0, \end{aligned}$$

that is, $\ell((ST^2)^k\, ST^x\, B) \geq \ell((ST^2)^{k+1}\, B)$. Moreover,

$$\begin{aligned} |\tilde{b}_{1,x}|^2 - |\tilde{b'}_{1,-2}|^2 &= [(\alpha(x+m) + \beta)^2 - (\alpha(m-2) - \beta)^2]|b_1^*|^2 \\ &= [\alpha(x-2+2m)(\alpha(x+2) + 2\beta)]|b_1^*|^2 \geq 0 \end{aligned}$$

and

$$|\tilde{b}_{2,x}|^2 - |\tilde{b'}_{2,-2}|^2 = [(\beta(x+m)+\gamma)^2 - (-\beta(m-2)+\gamma)^2]|b_1^*|^2$$
$$= [\beta(x-2+2m)(\beta(x+2)+2\gamma)]|b_1^*|^2 \geq 0,$$

that is, $\ell((ST^2)^k\, ST^x\, B) \geq \ell((ST^{-2})^{k+1}\, B)$. The second part of the proof is very similar. □

Proofs of Lemmas 9 and 10 are very similar to the proof of Lemma 8.

Lemma 9. *Let B be a proper basis and let x be an integer. Then $\ell(T^x\, B) \geq \ell(B)$.*

Lemma 10. *Let R be a reduced basis, let k be a positive integer, let x and x' be integers. Then*

- *if $x \geq x' \geq 0$, then $\ell((ST^2)^k\, ST^x\, R) \geq \ell((ST^2)^k\, ST^{x'}\, R)$;*
- *if $x \leq x' \leq 0$, then $\ell((ST^{-2})^k\, ST^x\, R) \geq \ell((ST^{-2})^k\, ST^{x'}\, R)$.*

Moreover, the following properties hold:

- *if m is positive, then $\ell((ST^2)^k\, S\, R) \geq \ell((ST^{-2})^k\, S\, R)$;*
- *if m is negative, then $\ell((ST^{-2})^k\, S\, R) \geq \ell((ST^2)^k\, S\, R)$.*

Proof. Proof of Theorem 5.

Notice first that, by Lemma 9, we can suppose that $x_{k+1} = 0$. Suppose that $k = 1$. Then, by Lemma 10, $\ell(ST^{x_1}B) \geq \ell(SB)$.

Suppose that $k \geq 1$. The variable ε will denote an integer in $\{1, -1\}$. We construct a sequence ω_0, ω_1, $\ldots$, ω_k such that the following properties hold:

(1) $\omega_0 = \omega$ and $\omega_k = (ST^{\varepsilon 2})^{k-1}S$;
(2) $\omega_j = (ST^{\varepsilon 2})^j \prod_{i=1}^{k-j} ST^{x_i}$, with $\varepsilon \in \{1, -1\}$ such that $(\varepsilon 2)x_{k-j} \geq 0$, for all $j \in \{1, \ldots, k-1\}$;
(3) $\ell(\omega_j\, B) \leq \ell(\omega_{j-1}\, B)$, for all $j \in \{1, \ldots, k\}$.

Notice that ω_j may be equal to ω_{j-1} for some $j \in \{1, \ldots, k\}$. We proceed by induction on an integer j such that $1 \leq j \leq k$. Put $\omega' = \prod_{i=1}^{k-1} ST^{x_i}$, $\tilde{B}' = \omega' B$ and $\tilde{B} = ST^{x_k}\tilde{B}' = \omega\, B$. By Lemma 7, we know that $\ell(\tilde{B}) = \ell(ST^{x_k}\tilde{B}') \geq \ell(ST^{\varepsilon 2})\tilde{B}')$. Now, either x_{k-1} is positive and we put $\omega_1 = ST^2\omega'$, or x_{k-1} is negative and we put $\omega_1 = ST^{-2}\omega'$, so that ω_1 is Gaussian.

Let j be an integer in $\{1, \ldots, k-2\}$ such that ω_j verifies (2) and (3). By Lemma 8, it is obvious that we can put:

- $\omega_{j+1} = (ST^2)^{j+1} \prod_{i=1}^{k-j-1} ST^{x_i}$ if x is positive;
- $\omega_{j+1} = (ST^{-2})^{j+1} \prod_{i=1}^{k-j-1} ST^{x_i}$ if x is negative.

Suppose now that we constructed $\omega_{k-1} = (ST^{\varepsilon 2})^{k-1}ST^{x_1}$. Then, by Lemma 10, we can put $\omega_k = (ST^2)^{k-1}S$ whether m is positive and $\omega_k = (ST^{-2})^{k-1}S$ whether m is negative. □

Appendix C. The maximum number of steps of the Gaussian algorithm

Proof. Proof of Theorem 6.

From the last section , we know that the input with the smallest length demanding k steps is $(ST^2)^k$ or $(ST^{-2})^k$. Let Q be equal to (ST^2). The symmetrical matrix Q

$$\begin{pmatrix} 2 & 1 \\ 1 & 0 \end{pmatrix}$$

has its eigenvalues equal to $1+\sqrt{2}$ and $1-\sqrt{2}$. It can be diagonalized and one deduces that there exist two constants $\alpha > 0$ and $\beta > 0$ such that all coefficients of Q^k are expressed in the form

$$\alpha(1+\sqrt{2})^k + \beta(1-\sqrt{2})^k = \alpha(1+\sqrt{2})^k \left(1 + \frac{\beta(1-\sqrt{2})^k}{\alpha(1+\sqrt{2})^k}\right).$$

This leads to the lower–bound proposed by the theorem. □

Proof. Proof of Corollary 1. □

Appendix D. Sorting algorithms

Lemma 11. *The rewriting process always terminates.*

Proof. Let $l\colon \Sigma^* \to \mathbb{N}$ be the map given by the rule

$$l(\omega) = \sum_{i=1}^{|\omega|} \alpha_i,$$

where $\omega = t_{\alpha_1} \dots t_{\alpha_{|\omega|}}$. Let $h\colon \Sigma^* \to \mathbb{N}$ be the map given by the rule

$$h(\omega) = \sum_{i=1}^{s(\omega)} (s(\omega) - i)(\max(\omega) - |\omega_i|),$$

where $[\omega_1, \dots, \omega_m]$ is the increasing decomposition of ω. We proceed here by induction on the nonnegative integers $l(\omega)$ and $h(\omega)$. Suppose that $l(\omega) = 0$. Then $\sigma = 1$ and the rewriting process is over. Suppose that $l(\omega) = 1$ and $h(\omega) = 0$. Then $\omega = t_1$ and the rewriting process is over. Notice that there exists no word on Σ^* such that $l(\omega) = 1$ and $h(\omega) > 0$.

Suppose now that $l(\omega) = l$, that $h(\omega) = h$ and that we can apply a rewriting rule. If we apply one of the rules (6.3) or (6.4), then we get a word ω' such that $l(\omega') < l(\omega)$. If we apply (6.5) or (6.6), then $l(\omega') = l(\omega)$. Let $[\omega_1, \dots, \omega_m]$ be the increasing decomposition of ω.

Suppose that we can apply (6.5). Then there exists $j \in \{1, \dots, m-1\}$ such that $\min(\omega_{j+1}) = i$ and $\min(\omega_j) > i + 1$. Put

$$\omega = \omega_1 \dots \omega_j \, t_i \, \omega'_{j+1} \dots \omega_m;$$
$$\omega' = \omega_1 \dots t_i \, \omega_j \, \omega'_{j+1} \dots \omega_m;$$

where ω'_{j+1} is such that $\omega_{j+1} = t_i \, \omega'_{j+1}$. Then

$$\begin{array}{rcll} h(\omega') & = & h(\omega) & -(s(\omega) - j)(\max(\omega) - |\omega_j|) \; -(s(\omega) - (j+1))(\max(\omega) - |\omega_{j+1}|) \\ & & & +(s(\omega) - j)(\max(\omega) - (|\omega_j| + 1)) \\ & & & +(s(\omega) - (j+1))(\max(\omega) - (|\omega_{j+1}| - 1)) \\ & = & h(\omega) & -(s(\omega) - j) + (s(\omega) - (j+1)) \\ & = & h(\omega) & -1. \end{array}$$

Suppose now that we can apply the rule (6.6). Then there exist $p, q \in \{1, \dots, m\}$ and $i, j, k \in \{1, \dots, \max(\omega)\}$ such that one of the following cases occurs.

Case 1. Suppose that $i < k \leq \max(\omega_p)$.

Then

$$\omega = \omega_1 \dots \omega'_p \, t_i \, t_k \, \omega''_p \dots \omega'_q \, t_j \, \omega''_q \dots \omega_m;$$
$$\omega' = \omega_1 \dots \omega'_p \, t_i \, t_j \, t_k \, \omega''_p \dots \omega'_q \, \omega''_q \dots \omega_m.$$

Case 2. Suppose that $i = \max(\omega_p)$ and $k = \min(\omega_{p+1})$.

Then,

$$\omega = \omega_1 \dots \omega'_p \, t_i \, t_k \, \omega'_{p+1} \dots \omega'_q \, t_j \, \omega''_q \dots \omega_m;$$
$$\omega' = \omega_1 \dots \omega'_p \, t_i \, t_j \, t_k \, \omega'_{p+1} \dots \omega'_q \, \omega''_q \dots \omega_m.$$

Thus,

$$\begin{array}{rcll} h(\omega') & = & h(\omega) & -(s(\omega) - p)(\max(\omega) - |\omega_p|) \; -(s(\omega) - q)(\max(\omega) - |\omega_q|) \\ & & & +(s(\omega) - p)(\max(\omega) - (|\omega_p| + 1)) \\ & & & +(s(\omega) - q)(\max(\omega) - (|\omega_q| - 1)) \\ & = & h(\omega) & -(s(\omega) - p) + (s(\omega) - q) \\ & = & h(\omega) & +p - q, \end{array}$$

which concludes the proof. □

Lemma 12. *Let ω be a reduced word, and let $[\omega_1, \dots, \omega_m]$ be its increasing decomposition. Let $p \in \{1, \dots, m-1\}$ be an integer. If t_i is in ω_{p+1}, then t_{i+1} is in ω_p.*

Proof. Let $p \in \{1, \dots, m\}$ be such that the property does not hold. Let $i \in \{1, \dots, \max(\omega)\}$ be the smallest integer such that $t_i \in \omega_{p+1}$, and $t_{i+1} \notin \omega_p$. Then, there exists ω'_p, ω''_p, ω'_{p+1}, $\omega''_{p+1} \in \Sigma^*$ such that

$$\omega = \omega_1 \dots \omega'_p \, \omega''_p \, \omega'_{p+1} \, t_i \, \omega''_{p+1} \dots \omega_m;$$

and such that the following inequalities hold:

$$\begin{aligned} &\max(\omega'_p) < i+1 < \min(\omega''_p); \\ &\max(\omega'_{p+1}) < i < \min(\omega''_{p+1}); \\ &\mathrm{Dist}(t_i, \omega''_p) > 1. \end{aligned}$$

Case 1. Suppose that $i = \min(\omega_{p+1})$.

Then $\omega'_{p+1} = 1$, ω''_p is maximal on the left and we can apply the rule (6.5).

Case 2. Suppose that $i \neq \min(\omega_{p+1})$ and that $t_{i-1} \in \omega''_{p+1}$.

Since $t_i \in \omega_p$ and $\mathrm{Dist}(t_i, \omega''_p) > 1$, we can apply the rule (6.4).

Case 3. Suppose that $i \neq \min(\omega_{p+1})$ and that $t_{i-1} \notin \omega''_{p+1}$.

Then $\mathrm{Dist}(t_i, \omega''_p \omega'_{p+1}) > 1$. Thus, either $\omega'_p = 1$ and we can apply the rule (6.5), or $\omega'_p \neq 1$ and we can apply the rule (6.6). □

The following corollary is an easy consequence of Lemma 12.

Corollary 4. *Let ω be a reduced word, and let $[\omega_1, \dots, \omega_m]$ be its increasing decomposition. Let $p \in \{1, \dots, m-1\}$ be an integer. Then*

(1) $\max(\omega_p) > \max(\omega_{p+1})$;
(2) $\mathrm{Dist}(\omega_p, \omega_{p+1}) \leq 1$.

The easy but tedious proof of the following lemma is left to the reader, who can proceed by induction.

Lemma 13. *Let $\omega = \omega_1 \dots \omega_m$ be a permutation put under increasing decomposition. Put $\omega[1, \dots, n] = [\alpha_1, \dots, \alpha_n]$ and $\omega_1 = \prod_{k=1}^{q} t_{i_k} \dots t_{i_k+p_k}$[7]. Suppose that ω is reduced. Then the following properties hold:*

(1) $\alpha_1 < \alpha_2 < \dots < \alpha_{i_1}$;
(2) $\alpha_{i_k} = \max\{\alpha_1, \alpha_2, \dots, \alpha_{i_k+p_k}\}$;
(3) $\alpha_{i_k} < \alpha_{i_k+p_k+1} < \alpha_{i_k+p_k+2} < \dots < \alpha_{i_{k+1}}$;

for all $k \in \{1, \dots, q\}$.

Corollary 5. *In the conditions of Lemma 13, the first iterations of the algorithm on $\omega[1, \dots, n]$ are represented by ω_1.*

The following corollary is an easy consequence of Lemma 13.

Corollary 6. *Let ω be a reduced word. Then ω is a bubblian word.*

Since the bubblian word associated to a given permutation is unique and the rewriting process terminates, the bubblian words are the reduced words.

D.1. **Insertion sort.** The basic idea of the insertion sort algorithm is the following: the beginning of the list being already sorted, the first non sorted element of the list is put at the right place in the already sorted part. Thus, an elementary transform made by the algorithm is a cycle $(i, i+1, \dots, i+p) = t_{i+p} \, t_{i+p-1} \dots t_{i+1} \, t_i$ in $\mathcal{S}$. Thus, any permutation σ can be written on the form

$$\sigma = \prod_{p=1}^{m} (i_p, \dots, i_p + q_p) = \prod_{p=1}^{m} t_{i_p+q_p} \dots t_{i_p},$$

and the word on Σ^* produced by the algorithm, which we will call *insertion word*, is a particular decomposition of σ. Let ω be a word on σ^*. The rewriting rules for the insertion sort algorithm are very similar to the rewriting rules for the bubble sort algorithm. In the following equations, i, j and k are arbitrary positive integers and ω is a word on Σ:

$$\text{if } \mathrm{Dist}(i, \omega) > 1, \text{ then } t_i \, \omega \, t_i \longrightarrow \omega; \tag{D.1}$$

[7] $\prod_{k=1}^{q} t_{i_k} \dots t_{i_k+p_k}$ is the consecutively increasing decomposition of ω_1.

$$\text{if } \mathrm{Dist}(i+1,\omega) > 1, \text{ then } t_{i+1}\, t_i\, \omega\, t_{i+1} \longrightarrow t_i\, t_{i+1}\, t_i\, \omega; \tag{D.2}$$

$$\text{if } \mathrm{Dist}(i+1,\omega) > 1, \text{ then } t_{i+1}\, \omega\, t_i \longrightarrow \omega\, t_{i+1}\, t_i; \tag{D.3}$$

$$\text{if } j-i > 1, \text{ then } t_{j+1}\, t_j\, t_i \longrightarrow t_{j+1}\, t_i\, t_j. \tag{D.4}$$

Let ω be a word on σ^*. Using similar methods that in Subsection 6.1, we can prove that, by applying repeatedly rules D.1 to D.4, we obtain the insertion word uniquely associated to ω.

D.2. **Selection sort.** The basic idea of the selection sort algorithm is the following: the beginning of the list being already sorted, the algorithm finds the smallest element of the non sorted list and put it at the right place at the end of the already sorted part. As in the previous subsection, an elementary transform made by the algorithm is a cycle $(i, i+1, \ldots, i+p)$, which can be written as $t_{i+p}\, t_{i+p-1} \ldots t_{i+1}\, t_i$ in Σ^*. The word on Σ^* produced by the algorithm, called *selection word*, is a particular decomposition of σ.

Let ω be a word on σ^*. The rewriting rules for the selection sort algorithm are very similar to the rewriting rules for the bubble sort algorithm. In the following equations, i, j and k are arbitrary positive integers and ω is a word on Σ:

$$\text{if } \mathrm{Dist}(i,\omega) > 1, \text{ then } t_i\, \omega\, t_i \longrightarrow \omega; \tag{D.5}$$

$$\text{if } \mathrm{Dist}(i,\omega) > 1, \text{ then } t_i\, \omega\, t_{i+1}\, t_i \longrightarrow \omega\, t_{i+1}\, t_i\, t_{i+1}; \tag{D.6}$$

$$\text{if } \mathrm{Dist}(i,\omega) > 1, \text{ then } t_{i+1}\, \omega\, t_i \longrightarrow t_{i+1}\, t_i\, \omega; \tag{D.7}$$

$$\text{if } i-j > 1, \text{ then } t_i\, t_j\, t_{j-1} \longrightarrow t_j\, t_i\, t_{j-1}. \tag{D.8}$$

Let ω be a word on σ^*. Using similar methods that in Subsection 6.1, we can prove that, by applying repeatedly rules D.1 to D.4, we can obtain the insertion word uniquely associated to ω.

Computer Science dept., University of Caen, F14032 Caen cedex, FRANCE
E-mail address: ali.akhavi@info.unicaen.fr
URL: http://www.info.unicaen.fr/~akhavi/
E-mail address: cmoreira@math.unicaen.fr
URL: http://www.math.unicaen.fr/~cmoreira/