COMPUTING GRÖBNER BASES ASSOCIATED WITH LATTICES

ISMARA ÁLVAREZ-BARRIENTOS, MIJAIL BORGES-QUINTANA AND MIGUEL ANGEL BORGES-TRENARD
Departamento de Matemática
Universidad de Oriente
Santiago de Cuba, Cuba

DANIEL PANARIO
School of Mathematics and Statistics
Carleton University
Ottawa, Canada

(Communicated by Eimear Byrne)

Abstract. We specialize Möller’s algorithm to the computation of Gröbner bases related to lattices. We give the complexity analysis of our algorithm. Then we provide experiments showing that our algorithm is more efficient than Buchberger’s algorithm for computing the associated Gröbner bases. Furthermore we show that the binomial ideal associated to the lattice can be constructed from a set of binomials associated with a set of generators of the corresponding label code. This result is presented in a general way by means of three ideal constructions associated with group codes that constitute the same ideal. This generalizes earlier results for specific cases of group codes such as linear codes, codes over \( \mathbb{Z}_m \) and label codes of lattices.

1. Introduction

Ideals associated to label codes of lattices are introduced in [2]. They are a generalization of the ideals associated to binary codes constructed from a generator matrix of the code [7]. The ideal associated to any linear code is introduced in [6] as the binomial ideal associated with the congruence that determines the code translated to certain monoid. In that article, Möller’s algorithm to compute Gröbner basis of these ideals is given; see also [5]. In this paper we focus on computing, efficiently, the Gröbner bases for these ideals. We provide a Möller type algorithm [16] for label codes of lattices. We implement our algorithm in GAP (version 4.7.2, [12]) and compared it with Buchberger’s algorithm in GAP. Experimental results with several lattices are also given.

There are three constructions of ideals associated with group codes. We prove in Theorem 3.3 that these constructions constitute the same ideal. This generalizes results for linear codes [6, 7], codes over \( \mathbb{Z}_m \) [14] and label codes of lattices [2]. In the latter case, our theorem entails that the binomial ideal associated to the lattice can be constructed from a set of binomials associated with a set of generators of the corresponding label code instead of using all the codewords as in [2]. On the
other hand, a construction given in the theorem allows the application of Möller’s algorithm to lattice coding. For this type of algorithms to compute Gröbner bases by means of linear algebra in a quotient algebra; some of the initial works on this topic are [10] and [11].

This paper is organized as follows. The general formulation of Möller’s algorithm is given in Section 2. Section 3 provides the required background on lattice coding and the associated ideal. Proposition 1 plays a central role in our construction of Möller’s algorithm for lattices. This algorithm is formulated, and its theoretical foundations explained, in Section 4. Computational results are described in Section 5. It is experimentally shown that, for ideals associated with label codes of lattices as defined in [2], our proposed solution for the computation of the reduced Gröbner bases is more efficient than Buchberger’s algorithm.

2. Möller’s algorithm

In this section we formulate Möller’s algorithm in a general commutative setting. We also comment on the most important concepts related to this algorithm.

Let $X = \{x_1, \ldots, x_n\}$ be a set of $n$ variables, $[X]$ the set of monomials in these variables (commutative free monoid), $\prec$ a term order on $[X]$, and $T_\prec(g)$ the maximal term of the polynomial $g$. The set of maximal terms of a set of polynomials $G$ with respect to $\prec$ is $T_\prec\{G\} = \{T_\prec(g) \mid g \in G \setminus \{0\}\}$. The set of maximal terms of an ideal $I$ is denoted by $T_\prec(I)$ and is the set of maximal terms of the polynomials that belong to the ideal. The $K$-vector space with basis $N_\prec(I) = [X] \setminus T_\prec(I)$, $\text{Span}_K(N_\prec(I))$, is the set of monomials in canonical form that determines $I$ with respect to $\prec$. When the context is clear we simply write $T(G)$, $T(I)$, $N(I)$. Given $a = (a_1, \ldots, a_n) \in \mathbb{Z}_+^n$, $x^a = \prod_{i=1}^n x_i^{a_i}$. For more details on these fundamental concepts see [1].

The next result characterizes the quotient algebra that determines an ideal.

**Theorem 2.1** ([1]). Let $\text{Span}_K(N(I))$ be the $K$-vectorial space with basis $N(I) = [X] \setminus T(I)$. Then,

i. $K[X] = I \oplus \text{Span}_K(N(I))$.

ii. For all $f \in K[X]$ there exists a unique polynomial in $\text{Span}_K(N(I))$, denoted by $\text{Can}(f, I)$, such that $f - \text{Can}(f, I) \in I$; moreover

- $\text{Can}(f, I) = \text{Can}(g, I)$ if and only if $f - g \in I$.
- $\text{Can}(f, I) = 0$ if and only if $f \in I$.

iii. For each $f \in K[X]$, $T(\text{Can}(f, I)) \preceq T(f)$.

iv. There exists an isomorphism of $K$-vectorial spaces between $\text{Span}_K(N(I))$ and $K[X]/I$ (this isomorphism associates $\text{Can}(f, I)$ with the class of $f$ modulo $I$).

**Definition 2.2.** Let $G \subseteq I$ be a set of polynomials generating the ideal $I$. Then, $G$ is a Gröbner basis of $I$ with respect to $\prec$ if $T_\prec\{G\}$ generates $T_\prec(I)$.

The Gröbner bases of an ideal $I$ have the key property that computing the remainder of the polynomial $f$ with respect to the set of polynomials that form the Gröbner basis is an algorithmic process with a unique result: $\text{Can}(f, I)$.

**Definition 2.3.** A Gröbner basis $G$ of $I$ is called a reduced Gröbner basis if for all $g \in G$

i. $T(g)$ is not a multiple of any $s \in T(G) \setminus \{T(g)\}$;

ii. the leading coefficient of $g$ is 1;

iii. $g = T(g) - \text{Can}(T(g), I)$. 
We assume that an ordering $\prec$ among the terms in $[X]$ has been fixed, that $I$ is an ideal that determines a finite dimensional quotient algebra (zero-dimensional ideal), and that the $K$-vectorial space $\text{Span}_K (N(I))$ is represented by

1. a $K$-vectorial space $E$ together with an effective function

\[ \text{LinearDependency}(v, \{v_1, v_2, \ldots, v_r\}] \text{ that for each finite set } \{v_1, v_2, \ldots, v_r\} \subset E \text{ of linearly independent vectors and each vector } v \in E, \] returns the value defined by

- \{\lambda_1, \lambda_2, \ldots, \lambda_r\} \subset K \text{ if } v = \sum_{i=1}^{r} \lambda_i v_i, \text{ or }
- \text{False} \text{ if } v \text{ is not a linear combination of } \{v_1, v_2, \ldots, v_r\};

2. an injective linear morphism $\xi : \text{Span}_K (N(I)) \rightarrow E$.

The following functions are used in the algorithm:

- \text{InsertNexts}[t, \text{List}] \text{ that inserts properly the products } tx \text{ (for } x \in X) \text{ in } \text{List}, \text{ keeping the terms in increasing order with respect to the term order } \prec. \text{ (We comment that } \text{InsertNexts} \text{ does not keep duplications, instead this function counts the number of times that an element } t \text{ has been inserted in } \text{List})

- \text{NextTerm}[\text{List}] \text{ eliminates the first element of } \text{List} \text{ and returns it as the result.}

\textbf{Algorithm 1. Möller’s Algorithm}

Input: $\prec$, a term order on $[X]$; $\xi : \text{Span}_K (N_\prec (I)) \rightarrow E$, a convenient representation of $\text{Span}_K (N_\prec (I))$ as previously specified.

Output: $rGb(I, \prec)$, the reduced Gröbner basis of $I$ with respect to $\prec$.

1. $G := \emptyset$; \text{List} := [1]; $N := \emptyset$; $r := 0$;
2. while \text{List} $\neq \emptyset$ do
3. \hspace{0.5cm} $t := \text{NextTerm}[\text{List}]$;
4. \hspace{0.5cm} if $t \notin T_\prec (G) \setminus T_\prec \{G\}$ then
5. \hspace{1cm} $v := \xi(\text{Can}(t, I, \prec))$;
6. \hspace{1cm} $\lambda := \text{LinearDependency}(v, \{v_1, v_2, \ldots, v_r\})$;
7. \hspace{1cm} if $\lambda \neq \text{False}$ then $G := G \cup \{t - \sum_{i=1}^{r} \lambda_i t_i\}$ (where $\lambda := (\lambda_1, \ldots, \lambda_r)$)
8. \hspace{1cm} else $r := r + 1$; $v_r := v$; $t_r := t$; $N := N \cup t_r$;
9. \hspace{1cm} \text{List} := \text{InsertNexts}[t_r, \text{List}]$;
10. \hspace{0.5cm} \text{Return}[G].

We recommend [15] and [16] to read about Möller’s algorithm, its theoretical foundations, and a description of several different contexts where it has been successfully used. Also, in [8] it is formulated a general pattern algorithm for a non commutative setting and for monoid and group algebras. Möller’s algorithm is based on a fundamental property already present in [9] that indicates that a nonzero polynomial belongs to an ideal if and only if the monomials of the polynomial are linearly dependent in the quotient algebra that determines the ideal. This is precisely what is done in Step 6 where the $v_i’s$ are the representation in $E$ of the canonical forms $t_i’s$ already constructed by the algorithm. Hence, if $v$ is linearly dependent from the $v_i’s$ we obtain a new polynomial for the Gröbner basis. If, on the contrary, $v$ is linearly independent of the $v_i’s$ we obtain a new canonical form with the monomial $t$. This new canonical form generates more elements to be included to the list $\text{List}$; the set that is inserted is $\{tx : x \in X\}$. We observe that all the elements $w \in \text{List}$ are such that there exist $x \in X$ and $v \in N$ such that $w = vx$. Due to this property of the elements of $\text{List}$ and to the condition in Step 4, we can guarantee that the Gröbner basis obtained is in reduced form. In fact, if the intention is to use this algorithm only to compute the structures known as “border basis” or only the Gröbner representation for the ideal, Step 4 should be eliminated.
Since the ideal $I$ is zero-dimensional, once the algorithm constructs a basis of the quotient $K[X]/I$ in $N$ no more elements are included in $List$, and so $List$ decreases through the Steps 3 to 7 until it is empty and the algorithm stops.

3. **Group codes associated with lattices**

3.1. **Lattices.** Let $\mathbb{R}^m$ be the real $m$-dimensional vector space together with the usual scalar product $\langle \cdot, \cdot \rangle$ and Euclidean norm $\|x\| = (x, x)^{1/2}$.

**Definition 3.1.** A lattice is a discrete additive subgroup $\Lambda$ of $\mathbb{R}^m$.

Every lattice is generated by integer combinations of some linearly independent vectors (lattice elements) $\{b_1, \ldots, b_n\} \subset \mathbb{R}^m$

$$\Lambda = \{ k_1 b_1 + \cdots + k_n b_n \mid k_i \in \mathbb{Z}\}.$$  

The set of vectors $b_1, \ldots, b_n$ is a basis of the lattice and $n$ is the dimension (or range) of $\Lambda$. Let $b_i = (b_{i1}, \ldots, b_{im})$, $1 \leq i \leq n$. Then, the generator matrix of $\Lambda$ is defined by $B = [b_1, \ldots, b_n]^t$, and the lattice can be described as

$$\Lambda = \{ v = xB \mid x \in \mathbb{Z}^n \}.$$  

The determinant (or volume) of $\Lambda$ is defined as $(BB^t)^{1/2}$. If $m = n$ the lattice has complete range and the volume of $\Lambda$ is $|\det(B)|$.

3.2. **Label code of a lattice.** A sequence space $G$ is the cartesian product of $n$ sets $G_i$, $1 \leq i \leq n$. A block code $C$ is any subset of a sequence space $G = G_1 \times \cdots \times G_n$. The codewords are the elements of the code. If each $G_i$ of the sequence space has group structure, then the subgroup $C$ of the sequence space group $G$ is a group code. Without loss of generality, each $G_i$ can be taken such that $G_i \cong \mathbb{Z}_{g_i}$. For more details on these concepts see [2, 4].

Consider the nested sequence of vector spaces $\{0\} \subset V_0 \subset V_1 \subset \cdots \subset V_n = \mathbb{R}^n$, where the dimension of $V_i$ as vector space is $i$. Let $W_i$ be the orthogonal complement of $V_{i-1}$ in $V_i$, that is, $V_i = V_{i-1} \oplus W_i$, with $1 \leq i \leq n$. Furthermore, let us assume that $\Lambda \subset V_n$ is an $n$-dimensional lattice. The ordered system of subspaces $S = \{ W_i \}_{i=1}^n$ corresponding to the sequence $\{ V_i \}_{i=1}^n$ is the coordinate system of $\Lambda$. For any subspace $V$ of $\mathbb{R}^n$, the cross section $\Lambda_V$ of $\Lambda$ is the set of lattice points that are in $V$, $\Lambda_V := \Lambda \cap V$. The cross sections $\Lambda_V$ have lattice structure. We denote the projection of the lattice to the vector space $W_i$ as $P_{W_i}(\Lambda)$.

The label group $G_i(\Lambda)$ of $\Lambda$ at level $i$, $1 \leq i \leq n$, is defined as the quotient group $P_{W_i}(\Lambda)/\Lambda_{W_i}$. In this way a label code of a lattice $L(\Lambda)$ is a subgroup of the sequence space group formed by the cartesian product of the $G_i(\Lambda)$’s.

The following proposition, that we use in our algorithm, provides a way to decide if a vector of the sequence space is in the code.

**Proposition 1 ([4]).** Let $G = \mathbb{Z}_{g_1} \times \cdots \times \mathbb{Z}_{g_n}$, and $c \in G$. Then $c \in L(\Lambda)$ if and only if

$$cP(\Lambda)V^{s^t} \in \mathbb{Z}^r,$$

where $V^*$ is a generator matrix of $\Lambda^*$ (the dual lattice of $\Lambda$), $r$ is the number of generators of $V^*$ (number of rows), $P(\Lambda) = \text{diag}(\det(P_{W_1}(\Lambda)), \ldots, \det(P_{W_n}(\Lambda)))$, where $\text{diag}(\cdot)$ is a diagonal matrix.

The matrix $H = P(\Lambda)V^{s^t}$ in the above proposition is the check matrix of the code.
3.2.1. Computing a set of generators for a lattice code. Let Λ be a lattice, \{v_1, \ldots, v_r\} be a generator set for Λ, L a label code, and \(C(\Lambda) = \text{diag}(\det(\Lambda v_1), \ldots, \det(\Lambda v_r))\). In a given coordinate system, \(v \in \Lambda\) can be written as \(v = kC(\Lambda) + cP(\Lambda)\) for \(c \in L\) and \(k \in \mathbb{Z}^n\); see [4]. We define the morphism \(\Phi : \Lambda \rightarrow L\) assigning to \(v \in \Lambda\) the codeword \(c \in L\). The set \(\{\Phi(v_1), \ldots, \Phi(v_r)\}\) is a generator set of \(L\); see [4], Section 3.D, p. 828.

3.3. Ideal associated with a group code. Let us assume that we have a sequence group \(G = \mathbb{Z}_{g_1} \times \cdots \times \mathbb{Z}_{g_n}\) and \(L\) a group code over \(G\).

Next we show several ways of introducing the same ideal associated with this group code in a more general setting than the ones in [7, 6] for binary codes and for linear codes in general, in [14] for codes over \(\mathbb{Z}_m\), and in [2] for label codes of lattices.

Depending on the context, an element will be treated as an integer or as an element in \(G\). For instance, in \(x^a\), \(a\) is a vector of positive integers, while in \(a \in G\) each component \(a_i\) is the corresponding element in \(G_i\). For \(a \in G\), \(x^a\) is the monomial such that the exponent of each variable is the corresponding \(a_i\) as an integer number, \(0 \leq a_i \leq g_i - 1\). This abuse of notation can be solved introducing two cross characteristic functions as in [14], however we decide not to do so in this paper since we need this only for this section.

Next we give three ways of associating an ideal with \(L\).

**Definition 3.2.** The ideal associated with the congruence that determines \(L\) over \(G\) is

\[
(1) \quad I_{\equiv_L} := \langle \{x^a - x^b : a, b \in \mathbb{Z}_+^n, a - b \in L\}\rangle.
\]

The ideal associated with the elements of \(L\) is

\[
(2) \quad I_{L_1} := \langle \{x^a - 1 : a \in L\} \cup \{x_i^{b_i} - 1 : i = 1, \ldots, n\}\rangle.
\]

Given \(L = \langle\{a_1, \ldots, a_k\}\rangle\), the ideal associated with this generator set of \(L\) is

\[
(3) \quad I_{L_2} := \langle \{x_i^{a_i} - 1 : i = 1, \ldots, k\} \cup \{x_i^{b_i} - 1 : i = 1, \ldots, n\}\rangle.
\]

The first is a natural way of introducing a binomial ideal associated with \(L\) using the equivalence relation that determines \(L\) in \(G\). The definition of an ideal associated with a linear code [6] is using \(I_{\equiv_L}\); this was later seen in [14] for codes over \(\mathbb{Z}_m\). On the other hand, \(I_{L_2}\) is used in [7] and [14] where is shown that is the same ideal as \(I_{\equiv_L}\). In [2] \(I_{L_1}\) is used to define the ideal associated with a lattice by means of the label code of the lattice.

The following theorem shows that in the three cases previously studied the three ways of defining the ideal are equivalent. Moreover, the result is true for any ideal associated with a group code over \(G\), where \(G = \mathbb{Z}_{g_1} \times \cdots \times \mathbb{Z}_{g_n}\).

**Theorem 3.3.** Let \(G = \mathbb{Z}_{g_1} \times \cdots \times \mathbb{Z}_{g_n}\) and \(L\) a subgroup of \(G\). Then, the three ideals given in Definition 3.2 coincide.

**Proof.** We first show the equality between \(I_{L_1}\) and \(I_{L_2}\). Clearly, \(I_{L_2} \subset I_{L_1}\). Let \(a \in L\), then there exist nonnegative integers \(\beta_i, 1 \leq i \leq k\), such that \(a = \sum_{i=1}^{k} \beta_i a_i\).
Let \( l \) be such that \( 1 \leq l \leq k \) and \( \beta_l \neq 0 \). Then
\[
x^a - 1 = x^{\beta a} x^{\sum_{i=1}^{k} \beta_i a_i} - 1 = (x^{\beta a_1} - 1)(x^{\sum_{i=1, i \neq l}^{k} \beta_i a_i} - 1) + (x^{\beta a_1} - 1) + (x^{\sum_{i=1}^{k} \beta_i a_i} - 1).
\]
We observe that the previous equality gives a recursion in the number of nonzero \( \beta \)'s, hence to prove that \( x^a - 1 \in I_{L_2} \) is sufficient to show that \( x^{\beta a_1} - 1 \in I_{L_2} \).

Obviously, \( x^{\beta a_i} - 1 \in I_{L_2} \), thus applying a recursion similar to above we get
\[
x^{\beta a_1} - 1 = (x^a - 1)(x^{(\beta_1-1)a} - 1) + (x^a - 1) + (x^{(\beta_1-1)a_1} - 1),
\]
from which we obtain \( x^{\beta a_1} - 1 \in I_{L_2} \).

We show now the equality between \( I_{L_1} \) and \( I_{\equiv L} \). In this case it is clear that \( I_{L_1} \subset I_{\equiv L} \); we prove the other inclusion. Let \( x^a - x^b \) with \( a, b \in \mathbb{Z}_+^n \) and \( a - b \in L \). Let \( g = (g_1, \ldots, g_n) \),
\[
\tag{4} a - b \in L \quad \text{if and only if} \quad a + (g - 1) b \in L.
\]
Then \( a + (g - 1) b \in L \) implies that there exists \( c \in L \), \( 0 \leq c \leq g \), such that
\[
a + (g - 1) b = c + \sum_{i=1}^{n} \alpha_i g_i, \quad \alpha_i \in \mathbb{Z}, \quad i = 1, \ldots, n.
\]

We decompose \( \alpha \) in a positive part and a negative part, \( \alpha = \alpha^+ - \alpha^- \) and substituting back in Equation (5), we get
\[
a + (g - 1) b + \sum_{i=1}^{n} \alpha^- g_i = c + \sum_{i=1}^{n} \alpha^+ g_i.
\]
We note that \( x^c - 1 \in I_{L_1} \) and \( x^{\alpha^+} g - 1 \in I_{L_1} \), then \( x^{c + \alpha^+} g - 1 \in I_{L_1} \) since
\[
x^{c + \alpha^+} g - 1 = (x^c - 1)(x^{\alpha^+} g - 1) - (x^{\alpha^+} g - 1) - (x^c - 1).
\]
Hence, using (6), \( x^{\alpha^+} g - 1 \in I_{L_1} \) and from \( x^{\alpha^-} g - 1 \in I_{L_1} \) we obtain \( x^{a + (g - 1)b + \alpha^-} g - 1 \in I_{L_1} \). Furthermore, it is clear that \( x^{gb} - 1 \in I_{L_1} \). Therefore, \( x^a - x^b \in I_{L_1} \) since
\[
x^a - x^b = x^b (x^{a + (g - 1)b} - 1) - x^a (x^{gb} - 1),
\]
and so \( I_{\equiv L} \subset I_{L_1} \). \( \square \)

4. Möller’s Algorithm for Lattices

In order to specialize Möller’s algorithm to lattices it is necessary to define the objects in this context, as well as to understand the linear dependency in which Proposition 1 plays a crucial role. In Section 2, we give Algorithm 1 that presents a general formulation of Möller’s algorithm. Here we associate this algorithm with lattices using an additive monoid structure for the lattice and considering the application of Möller’s algorithm to linear codes; see [5].

4.1. Main Objects. The injective linear morphism \( \xi : [X] \to \mathbb{R}^n \) associates to each monomial a vector in \( \mathbb{R}^n \) as follows:

- let \( e_i \) be the \( i \)-th coordinate vector of \( G \), that is, \( e_{ij} = 1 \) if \( i = j \) and \( e_{ij} = 0 \) if \( i \neq j \);
- \( \xi(x_i) = \text{frac}(e_i P(A) V^* T) \), that is, \( \xi(x_i) = \text{frac}(e_i H) \), where \( \text{frac}(\cdot) \) represents the fractional part between 0 and 1 of the real number.
• \( \xi(\prod_{i=1}^{n} x_i^{m_i}) = \frac{\sum_{i=1}^{n} (m_i \mod g_i) \xi(x_i)\), that is, \\
\( \xi(\prod_{i=1}^{n} x_i^{m_i}) = \frac{\sum_{i=1}^{n} m_i \epsilon_i)H)\);

• let \( \psi : [X] \rightarrow G \) be a morphism of monoids such that \( \psi(x_i) = e_i \) extending the morphism of monoids naturally \( \psi(u) = \psi(\prod_{i=1}^{n} x_i^{m_i}) = \sum_{i=1}^{n} m_i \epsilon_i\); we note that the relation between \( \xi \) and \( \psi \) is given by \( \xi(u) := \psi(u)H \).

The equivalence relation \( R_L \) that determines the label code \( L \) over \( G \), can be translated to \( [X] \), given \( x^a, x^b \in [X], x^a \equiv_L x^b \) if and only if \( (\psi(x^a), \psi(x^b)) \in R_L \), that by Proposition 1 happens if and only if \( (\psi(x^a) - \psi(x^b))H \in \mathbb{Z}^n \), that is if the fractional parts of \( \psi(x^a) \) and \( \psi(x^b) \) are equal.

Given Theorem 3.3, we have that the ideal \( I_{L_1} \) associated to the lattice as in [2] is the ideal \( I_{\equiv_L} \). This is the ideal associated to the application of Möller’s algorithm. We denote this ideal by \( IL \). Having in mind [5] for the application of Möller’s algorithm to linear codes, the role of the ideal associated to the linear code is here played by \( IL \).

**Algorithm 2.** Möller’s algorithm for lattices

Input: \( \prec \), an ordering of the terms in \( [X] \); \( H \), control matrix of the code.

Output: \( rGB(IL, \prec) \), the reduced Gröbner basis of \( IL \) with respect to \( \prec \).

1. \( G := \emptyset; \ List := \{1\}; \ N := \emptyset; \ r = 0; \)
2. while \( \List \neq \emptyset \) do
3. \( \ w := \text{NextTerm}(\List); \)
4. \( \text{if} \ t \notin T_{\prec}(G) \setminus T_{\prec}(G) \) then
5. \( \ v := \xi(\text{Can}(t, I, \prec)); \)
6. \( \Lambda := \text{Member}[v, \{v_1, v_2, ..., v_r\}]; \)
7. \( \text{if} \ False \neq \Lambda \text{ then} \ G := G \cup \{w - w_j\}; \)
8. \( \text{else} \ r := r + 1; v_r := v; t_r := t; \ N := N \cup \{t_r\}; \)
9. \( \List := \text{InsertNexts}(t_r, \List); \)
10. return \( [G] \);

4.2. **Description of the main components of the algorithm.**

1. **InsertNexts**\( [t, \List] \): This function inserts the multiples of the monomial \( t \) given by \( \{t \cdot x : x \in X\} \) in \( \List \) following an ordering of the given terms. No element is repeated in the list but a count of the presence of each element is kept.

2. **Member**\( [v, \{v_1, v_2, ..., v_r\}]; \); This function substitute the function **LinearDependency** of Algorithm 1 returning
   a. \( j \) if \( v \) is in some of the equivalence classes already existing, \( v - v_j \in \mathbb{Z}^n \); this means that the fractional parts of \( v \) and \( v_j \) are equal in each component;
   b. \( False \) if \( v \) is not in any of the equivalence classes already existing, that is \( v - v_i \notin \mathbb{Z}^n \) with \( 1 \leq i \leq r \).

To search this element we proceed as in **InsertNext** but we use lexicographical order between the vectors \( v_j \) of the fractional parts.

3. To check the condition on Step 4 we use an strategy introduced in [11]. \( \List \) counts the quantity of times that \( t \) is inserted in the list; hence, \( t \) is not a maximal term, that is at the same time a multiple of the other maximal terms, if the number of times it was inserted in the list is equal to the number of variables belonging to the support of \( t \).

4. **NextTerm**\( [\List] \): This function removes the first element of the list returning two values: the removed element and the resulting list.
We next show that Algorithm 2 correctly computes the reduced Gröbner basis.

**Theorem 4.1.** Algorithm 2 computes a reduced Gröbner bases of the ideal of the label code $I_L$ corresponding to the matrix $H$ under the specified ordering $\prec$.

*Proof.* The correctness of this algorithm is a direct consequence of the application of Möller’s algorithm to the lattice context. It is important in this specialization to define the objects of the algorithm correctly; see Section 4.1. Then, the algorithm finishes given the finite dimension of the quotient vector space and so the result is a reduced Gröbner basis of the ideal that constitutes the kernel of the morphism $\xi$ that defines the equivalence relation. In the analysis in Section 4.1 this ideal is $I_L$.

**Complexity analysis.** For a complete complexity analysis see [11], [13] and [15]. Here we analyze the number of steps that the algorithm executes. The operations are in $G$, $\mathbb{Z}^n$ or computations of the fractional parts of the real numbers. The dimension of the quotient vector space is $d = \#(G)/\#(L) = (\prod_{i=1}^{n} g_i)/\#(L)$, where $\#(\cdot)$ denotes the cardinality of the set and the number of variables is $n$. Hence, the number of steps that the algorithm makes is $O(n^2d^2)$. We observe that the number of steps that the *while* step is executed is the quantity of elements that the *List* can have and is bounded by $nd$, while the cost inside the cycle is dominated by the calls to *Member* and *InsertNexxts*. The function *Member* performs an insertion in an ordered list of $r \leq d$ and *InsertNexxts* is a merging procedure between ordered lists of length $nd$ and $n$, and so the cost is dominated by $nd$.

We note that the parameter $d$ gives an exponential growth to these algorithms. Nevertheless, the efficiency of them is based on the fact that the quantity of elements that the algorithm analyzes to obtain the results is quadratic in relation to the dimension of the quotient vector space.

## 5. Experimental results

In this section we present our experimental results that we summarize in Table 1. In this table we show the running times of Möller’s algorithm to compute a reduced Gröbner basis with respect to the graded reverse lexicographical order as well as the time spent by Buchberger’s algorithm. We used GAP [12] on a Pentium Dual-Core CPU, 2.00 GHz, 2.00 GHz, 3 Gb RAM computer; the time is given in milliseconds. We consider the lattices given in Table I of [2], using the bases and coordinate systems as in [3] to obtain the minimal trellis diagrams\(^2\). To compute the Gröbner basis in Buchberger’s algorithm, we compute for each lattice code a generator set (see Section 3.2.1) and we construct the set of binomials that generate the ideal following (3) in Definition 3.2.

We remark that in all cases the running time of Möller’s algorithm is inferior to the one of Buchberger’s algorithm. This difference grows when the number of variables, the dimension of the ideal and the size of the reduced Gröbner bases grow.

## 6. Conclusions

In this work we show that there exist three equivalent ways of introducing an ideal associated with a group code for any finite Abelian group. One of these forms, the ideal associated with the equivalence relation that determines the finite Abelian

\(^2\)We note that the result for $E_6$ in Table 1 is different of the one in [2], because in this case in [2] it was not used the base and coordinate systems given in [3].
Reduced Gröbner bases associated with lattices

| Lattice | Reduced Gröbner basis | AM. (t) | AB. (t) |
|---------|------------------------|---------|---------|
| $D_4$   | $x_1^2 - 1$, $x_2x_3 - x_1x_4$, $x_2^2 - x_3x_4$, $x_1^2 - x_3 - x_4$, $x_1x_2^2 - x_2$, $x_2^2 - x_1 - x_3 - x_4$, $x_1x_2^2 - x_2$ | 15      | 16      |
| $A_4$   | $x_2x_4 - x_1$, $x_2x_4 - x_3$, $x_2^2 - x_4$, $x_2x_5 - x_4$, $x_2^2 - 1$, $x_1x_2 - x_3^2$, $x_1x_2^2 - 1$ | 15      | 16      |
| $A_4^*$ | $x_2x_4 - x_1$, $x_2x_4 - x_3$, $x_2^2 - x_4$, $x_2x_5 - x_4$, $x_2^2 - 1$, $x_1x_2 - x_3^2$, $x_1x_2^2 - 1$ | 16      | 62      |
| $A_5$   | $x_2^2 - 1$, $x_3x_4 - x_2x_5$, $x_3x_4 - x_3x_5$, $x_3^2 - 1$, $x_1x_2 - x_3^2$, $x_1x_2^2 - 1$, $x_2x_4x_5 - x_3$, $x_3x_5 - x_4$, $x_3x_5 - x_2$ | 31      | 94      |
| $A_5^*$ | $x_2^2 - x_4$, $x_3x_4 - x_1$, $x_2x_5 - x_3$, $x_2^2 - 1$, $x_3x_4 - x_1x_5$, $x_2x_4 - x_1$, $x_1x_2 - x_2^2$, $x_1^2 - 1$, $x_1x_2 - x_4$, $x_2^2 - 1$ | 15      | 47      |
| $E_6$   | $x_2^2 - 1$, $x_3x_4 - x_2x_5$, $x_3x_5 - x_3x_6$, $x_3^2 - 1$, $x_1x_2 - x_2x_5$, $x_2x_4 - x_1x_6$, $x_2x_4 - x_2x_6$, $x_3^2 - x_4x_6$, $x_2x_4 - 1$, $x_3x_4 - x_4x_6$, $x_2^2 - 1$ | 47      | 187     |
| $E_6^*$ | $x_2^2 - 1$, $x_3x_4 - 1$, $x_2x_5 - x_3x_6$, $x_3^2 - 1$, $x_1x_2 - 1$, $x_3x_4 - 1$, $x_2x_4 - 1$, $x_3x_4 - 1$, $x_1x_2 - 1$, $x_3x_4 - 1$, $x_2x_4 - 1$ | 47      | 125     |
| $E_7$   | $x_2^2 - 1$, $x_3x_4 - x_2x_5$, $x_3x_5 - x_3x_6$, $x_3^2 - 1$, $x_1x_2 - x_2x_5$, $x_2x_4 - x_2x_6$, $x_3^2 - x_4x_6$, $x_2x_4 - 1$, $x_3x_4 - x_4x_6$, $x_2^2 - 1$ | 62      | 235     |
| $E_7^*$ | $x_2^2 - 1$, $x_3x_4 - 1$, $x_2x_5 - x_3x_6$, $x_3^2 - 1$, $x_1x_2 - 1$, $x_3x_4 - 1$, $x_2x_4 - 1$, $x_3x_4 - 1$, $x_1x_2 - 1$, $x_3x_4 - 1$, $x_2x_4 - 1$ | 31      | 94      |
| $E_8$   | $x_2^2 - 1$, $x_3x_4 - 1$, $x_2x_5 - x_3x_6$, $x_3^2 - 1$, $x_1x_2 - 1$, $x_3x_4 - 1$, $x_2x_4 - 1$, $x_3x_4 - 1$, $x_1x_2 - 1$, $x_3x_4 - 1$, $x_2x_4 - 1$ | 88      | 1054    |

Table 1. Reduced Gröbner bases for several lattices.

References

[1] W. W. Adams and P. Loustaunau, An Introduction to Gröbner Bases, Amer. Math. Soc., 1994.
[2] M. Aliasgari, M. R. Sadeghi and D. Panario, Gröbner bases for lattices and an algebraic decoding algorithm, IEEE Trans. Commun., 61 (2013), 1222–1230.
[3] A. H. Banihashemi and I. F. Blake, Trellis complexity and minimal trellis diagrams of lattices, IEEE Trans. Inform. Theory, 44 (1998), 1829–1847.
[4] A. H. Banihashemi and F. R. Kschischang, Tanner graphs for group block codes and lattices: construction and complexity, IEEE Trans. Inform. Theory, 47 (2001), 822–834.

[5] M. Borges-Quintana, M. A. Borges-Trenard and E. Martínez-Moro, A Gröbner representation for linear codes, in Adv. Coding Theory Crypt., World Scientific, 2007, 17–32.

[6] M. Borges-Quintana, M. A. Borges-Trenard and E. Martínez-Moro, On a Gröbner bases structure associated to linear codes, J. Discrete Math. Sci. Crypt., 10 (2007), 151–191.

[7] M. Borges-Quintana, M. A. Borges-Trenard, P. Fitzpatrick and E. Martínez-Moro, On Gröbner basis and combinatorics for binary codes, Appl. Algebra Engin. Commun. Comp., 19 (2008), 393–411.

[8] M. A. Borges-Trenard, M. Borges-Quintana and T. Mora, Computing Gröbner bases by FGLM techniques in a non-commutative setting, J. Symb. Comput., 30 (2000), 429–449.

[9] B. Buchberger, An Algorithm for Finding a Basis for the Residue Class Ring of a Zero-dimensional Ideal (in German), Ph.D. thesis, Univ. Innsbruck, Austria, 1965.

[10] B. Buchberger and H. M. Möller, The construction of multivariate polynomials with preassigned zeros, in EUROCAM ’82, Marseille, 1982, 24–31.

[11] J. C. Faugere, P. Gianni, D. Lazard and T. Mora, Efficient computation of zerodimensional Gröbner bases by change of ordering, J. Symb. Comput., 16 (1993), 329–344.

[12] The GAP Group, GAP–Groups, Algorithms, and Programming, available online at http://www.gap-system.org

[13] S. Lundqvist, Complexity of comparing monomials and two improvements of the BM-algorithm, in Math. Methods Computer Science, Springer, Berlin, 2008, 105–125.

[14] I. Márquez-Corbella and E. Martínez-Moro, Algebraic structure of the minimal support codewords set of some linear codes, Adv. Math. Commun., 5 (2011), 233–244.

[15] T. Mora, Solving Polynomial Equation Systems II: Macaulay’s Paradigm and Gröbner Technology, Cambridge Univ. Press, 2005.

[16] T. Mora, The FGLM problem and Möller’s algorithm on zero-dimensional ideals, in Gröbner, Coding, and Cryptography, Springer, 2009, 379–384.

Received for publication March 2015.

E-mail address: ismara.alvarez@uo.edu.cu
E-mail address: mijail@uo.edu.cu
E-mail address: mborges@uo.edu.cu
E-mail address: daniel@math.carleton.ca