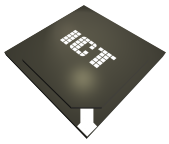
TUHH

Technische Universität Hamburg-Harburg

Gröbner Fans for Linear Codes

Institute of Computer Technology



Bachelor Thesis

submitted by

Daniel Rembold
born on 27.06.1992 in Hamburg

July 14, 2014

advised by: Dipl.-Technomath. Natalia Dück
supervised by: Prof. Dr. Dr. habil. Karl-Heinz Zimmermann

Abstract

This work is about...

Zusammenfassung

In dieser Arbeit geht es um...

Ackknowledgements

I would like to thank....

Danksagung

Ich möchte mich bei xy bedanken...

Statutory Declaration

Eidesstaatliche Erklärung

Hiermit versichere ich, die vorliegende Abschlussarbeit selbstständig und nur unter Verwendung der von mir angegebenen Quellen und Hilfsmittel verfasst zu haben. Sowohl inhaltlich als auch wörtlich entnommene Inhalte wurden als solche kenntlich gemacht. Die Arbeit hat in dieser oder vergleichbarer Form noch keinem anderem Prüfungsgremium vorgelegen.

Datum: _____ Unterschrift: _____

Contents

1 Introduction

1.1 Motivation

1.2 Tasks

1.3 Structure

2 Mathematical Background

In this chapter a mathematical basis is systematically approached to give the reader an understanding to Groebner Bases and Groebner fans.
In the first section monomials are revisited. The second section explains how monomials can be totally ordered. After that Ideals are defined over polynomial rings and a summary on Groebner bases and Groebner fans for ideals is presented. Furthermore, the next sections deal with enumerating Groebner bases on special ideals and finally, linear codes are presented and the connection between Groebner bases and the linear codes.

2.1 Monomials

In this section a brief explanation of polynomials are given.

Definition 2.1 (Monomial) *A monomial m is a product of variables over a finite field $\mathbb{K}$, denoted by $\mathbb{K}[X_1, X_2, \cdots X_n]$ of the form $X_1^{u_1} X_2^{u_2} \cdots X_n^{u_n}$, where $u_i, 1 < i < n$ and $u \in \mathbb{N}_0$*

The total **degree** of a monomial is $deg(m) = \sum_{i=1}^{n} u_i$

Definition 2.2 (Polynomial) *A polynomial f is a finite linear combination with coefficients $c_u \in \mathbb{K}$ multiplied with monomials.*

$$f = \sum_{u} c_u X^u$$

If $c_u \neq 0$ then $c_u x_u$ is a term of f.

2.2 Monomial Order

It is necessary to rearrange a polynomial with respect to a monomial order. That forms the foundation for dividing polynomials in the finite field $\mathbb{K}[X_1, X_2, \cdots X_n]$ and solving the Ideal membership problem.

Definition 2.3 (Term Ordering) *[1] A monomial order is a relation $>$ on the set of all monomials in $\mathbb{K}[x]$. Let m_1, m_2 and m_3 be monomials*

- *for any pair of monomials m_1,m_2, either $m_1 > m_2$ or $m_2 > m_1$ or $m_1 = m_2$*
- *if $m_1 > m_2$ and $m_2 > m_3$ then $m_1 > m_3$*
- *$m_1 > 1$ for any monomial $m_1 \neq 1$*
- *if $m_1 > m_2$ then $m \cdot m_1 > m \cdot m_2$ for any monomial m*

Two commonly used term orders are the following. Let u and v be elements of $\mathbb{N}_0^n$

Lexicographic Order[1] $u >_{lex} v$ if in $u - v$ the left most non-zero entry is positive. This can be written as $X^u >_{lex} X^v$ if $u >_{lex} v$.

Graded Lex Order[1] $u >_{grlex} v$ if $deg(u) > deg(v)$ or if $deg(u) = deg(v)$ and $u >_{lex} v$

Example 1 Let $m_1 = 4x^2y^4z^3$ and $m_2 = x^1y^1z^4 \in \mathbb{K}[x, y, z]$. The monomials can also be written as $m_1 = X^{(2\,4\,3)}$ and $m_2 = X^{(1\,1\,4)}$. Thus $m_1 >_{lex} m_2$ because the left most non-zero entry of $(2\ 4\ 3) - (1\ 1\ 4)$ is positive.

The total degree of m_1 is 9 and $deg(m_2) = 6$. Hence, $m_1 >_{lex} m_2$ and $deg(m_1) > deg(m_2)$ so that $m_1 >_{grlex} m_2$

$\Diamond$

Weight vectors

In order to compare monomials with a generic vector $(a_1, \ldots, a_n) \in \mathbb{R}^n_{\geq 0}$, the dot product with the exponent vector has to be build. The highest result is the leading term. If a tie occurs, some other fixed monomial order has to be used. Note that the standard monomial orders can be expressed as weight vector. The lexicographic order needs for instance all canonical unit vectors.

Leading term

Given a term order $>$, each non-zero polynomial $f \in \mathbb{K}[x]$ has a unique leading term, denoted by $\mathrm{LT}(f)$, given by the largest involved term with respect to the term order.
If $\mathrm{LT}(f) = cX^u$, where $c \in \mathbb{K}$, then c is the leading coefficient of f and X^u is the leading monomial(lm) or the initial monomial .[2]

Example 2 Let $f = 3x^2y^5z^3 + x^4 - 2x^3y^4 + 12x^2z^2$
With respect to lex order : $f = \underline{x^4} - 2x^3y^4 + 3x^2y^5z^3 + 12x^2z^2$
with respect to grlex order : $f = \underline{3x^2y^5z^3} - 2x^3y^4 + x^4 + 12x^2z^2$
with respect to the weight vector $(3, 2, 1)$: $f = \underline{3x^2y^5z^3} - 2x^3y^4 + x^4 + 12x^2z^2$
The underlined terms are the leading terms with the respect to the monomial order.

2.3 Ideals

Definition 2.4 (Ideal) *[2] An ideal I is collection of polynomials $f_1, \cdots, f_s \in \mathbb{K}[X_1, \cdots, X_n]$ and polynomials which can be built from these with multiplication with arbitrary polynomials and linear combination.*

This is called an Ideal generated by $f_1, \cdots, f_s$

$$\langle f_1, \cdots, f_s \rangle = \left\{ \sum_{i=1}^{s} h_i f_i \mid h_1, \cdots, h_s \in \mathbb{K}[X_1, \cdots, X_n] \right\}$$

It satisfies [1]:

- $0 \in I$
- If $f, g \in \langle f_1, \cdots, f_s \rangle$,then $f + g \in \langle f_1, \cdots, f_s \rangle$
- If $f \in \langle f_1, \cdots, f_s \rangle$ and $h \in \langle f_1, \cdots, f_s \rangle$, then $f \cdot h \in \langle f_1, \cdots, f_s \rangle$

Example 3 Let $I = \langle f_1, f_2 \rangle = \langle x^2+y, x+y+1 \rangle$ and $f = yx^2+y^2+x^2+xy+x$. Since $f = y \cdot f_1 + x \cdot f_2, f \in I$

$\Diamond$

Definition 2.5 (Initial Ideal) *[3]The initial Ideal of I with respect to a term order* $>$ *on* $\mathbb{K}[X_1, \cdots, X_n]$ *is the monomial ideal*

$$in_>(I) = \langle in_>(f) : f \in I \rangle$$

$in_>(f)$ means the leading term of f with respect to $>$. In other words, the initial ideal of I is the ideal generated by its leading terms.

2.4 Division Algorithm

The Ideal membership problem is easy to solve in a one-dimensional polynomial ring. It is only necessary to make the polynomial division and check if the remainder is zero. If result has no remainder, the polynomial p lies in the ideal I. But in a ring with several variables like $\mathbb{K}[X_1, X_2, \cdots X_n]$, the usual division algorithm will not work. A generalized algorithm is needed.

The goal is to divide $g \in \mathbb{K}[X_1, \cdots, X_n]$ by $f_1, \ldots, f_s \in \mathbb{K}[X_1, \cdots, X_n]$, so g can be expressed in the form

$$g = a_1 f_1 + \ldots + a_s f_s + r$$

where the $a_1 f_1 + \ldots + a_s f_s$ and $r \in \mathbb{K}[X_1, \cdots, X_n]$ This is possible with the Theorem mentioned at [4]. The remainder r is zero or r is a linear combination, with the coefficients in $\mathbb{K}$, none of which is divisible by any of $\mathrm{LT}(f_1), \ldots, \mathrm{LT}(f_s)$. Furthermore, if $a_i f_i \neq 0$, then $deg(g) \geq deg(a_i f_i)$

Algorithm 1 Division Algorithm [2]

Require: Basis $\langle f_1, \cdots, f_m \rangle$ of nonzero polynomials
Ensure: $r = 0$ or none of the terms in r are divisible by $LT_{\leq}(f_1), \cdots, LT_{\leq}(f_m)$

```
h_1 ← 0, ··· , h_m ← 0; r ← 0; s ← f
while s ≠ 0 do
    i ← 1
    division_occured ← false
    while i ≤ m and division_occured = false do
        if LT(f[i]) divides LT(s) then
            s ← s − (LT(s)/LT(f[i])) * f_i
            h_i ← h_i + LT(s)/LT(f_i)
            division_occured = true
        else
            i ← i + 1
        end if
    end while
    if division_occured = false then
        r ← r + LT(s)
        S ← s − LT(s)
    end if
end while
```

Example 4 Consider the ideal $I = \langle f_1, f_2 \rangle = \langle xy^2 + z, y^2 - 1 \rangle$ and the polynomial $f = x^3y^2 + x^2z$. First, with respect to the lex-order, applying the division the expression : $f = x^2(xy^2 + z) + 0(y^2 - 1) + 0$
But the division with f and $I = \langle f_2, f_1 \rangle$ gives the expression
$f = x^3(y^2 - 1) + x^2(xy^2 + z) - x^3y^2 + x^3$.

$\Diamond$

This example shows that is still possible to obtain a non-zero remainder even if $f \in \langle f_1, f_2 \rangle$. That means r = 0 is a necessary condition for the ideal membership but not a sufficient condition.

2.5 Groebner basis

The solution of the ideal membership problem needs a certain generating set for an Ideal I. It would be helpful if the remainder r on division is uniquely determined and the condition $r = 0$ is equivalent to the membership in the ideal.

Definition 2.6 (Groebner base) *[2] Let $\leq$ be a monomial order on $\mathbb{K}[X_1, \cdots, X_n]$ and let I be an Ideal on $\mathbb{K}[X_1, \cdots, X_n]$. A Groebner basis for I (with respect to $\leq$) is a finite set of polynomials $F = \{f_1, \ldots, f_m\}$ in I with the property that for every nonzero $f \in I, \mathrm{LT}_{\geq}(f\}$ is divisible by $\mathrm{LT}(f_i)$ for some $1 \leq i \leq m$*

A Groebner basis has the beneficial property that the remainder r of f by the elements of a Groebner basis are uniquely determined and independent of the order of the elements in G. Also every Ideal in $\mathbb{K}[X_1, \cdots, X_n]$ has a Groebner basis with respect to any monomial order[2].

In order to obtain a Groebner basis of an arbitrary basis $f_1, \ldots, f_n$ with an arbitrary monomial order $>$ of an Ideal I, an algorithm is needed. This algorithm is called Buchberger-Algorithm. The idea is to build every possible S-Polynomial of (f_i, f_j) for every $1 \leq i \neq j \leq n$ and every nonzero result is added to the basis until every S-Pair of (f_i, f_j) vanishes.
Let the polynomials $f, g \in \mathbb{K}[X_1, \cdots, X_n]$ and $\mathrm{LT}_{\leq}(f) = cX^{\alpha}$, $\mathrm{LT}_{\leq}(g) = dX^{\beta}$ and $\mathrm{LCM}\left(X^{\alpha}, X^{\beta}\right)$ be the least common multiple between X^{α} and X^{β}.

Definition 2.7 (S-Polynomial) *[2] The S-polynomial of f and g is the polynomial*

$$S(f,g) = \frac{\mathrm{LCM}\left(X^{\alpha}, X^{\beta}\right)}{\mathrm{LT}_{\leq}(f)} \cdot f - \frac{\mathrm{LCM}\left(X^{\alpha}, X^{\beta}\right)}{\mathrm{LT}_{\leq}(g)} \cdot g$$

Example 5 Consider the polynomials the polynomial ring $\mathbb{K}[x, y, z]$ with the basis $\{f, g\} = \{xy^2 - xz + y, xy - z^2\}$ with respect to the lexicographic order. Forming the S-Polynomial leads to:

$$\begin{aligned} S(f,\ g) &= \frac{\mathrm{LCM}(xy^2, xy)}{xy^2} \cdot \left(xy^2 - xz + y\right) - \frac{\mathrm{LCM}(xy^2,\ xy)}{xy} \cdot (xy - z) \\ &= \frac{xy^2}{xy^2} \cdot \left(xy^2 - xz + y\right) - \frac{xy^2}{xy} \cdot (xy - z) \\ &= -xz - yz + y \end{aligned}$$

$\Diamond$

The S-Polynomial is not zero and is not disvisible by the leading terms of f or g. That means the basis given in the example is not a Groebner basis. This can be deduced by the Buchberger criterium.

Definition 2.8 (Buchberger Criterion) *[2] A finite set $G = \{f_1, \cdots, f_m\}$ of polynomials in $\mathbb{K}[X_1, \cdots, X_n]$ is a Groebner basis of an Ideal $I = \langle f_1, \cdots, f_m \rangle$ if and only $S(f_i, f_j) = 0$, $\forall\ 1 \leq i, j \leq n, i \neq j$*

Algorithm 2 Buchbergers Algorithm $[KHZ]$

Require: Basis $F = (f_1, \cdots, f_m)$
Ensure: Groebner basis G for $I = \langle f_1, \cdots, f_m \rangle$ with $F \subseteq G$

```
G ← F
repeat
    G' ← G
    for each pair f_i and f_j in G , i ≠ j do
        S ← S(f_i, f_j)^{G'}          ▷ S-Polynomial with the basis of G'
        if G ≠ 0 then
            G ← G ∪ {S}
        end if
    end for
until G = G'
```

This algorithm is correct and terminates.[2]

However, a Groebner basis is not unique. A arbitrary polynomial can be added to a Groebner basis and will be remain a Groebner basis. Fortunately, each nonzero Ideal in $\mathbb{K}[X_1, \cdots, X_n]$ has a unique *reduced* Groebner basis with respect to a fixed monomial order.

Definition 2.9 (Reduced Groebner basis) *[2] A Groebner basis $G = \{f_1, \cdots, f_m\}$ in $\mathbb{K}[X_1, \cdots, X_n]$ is reduced if the polynomials $f_1, \cdots, f_m$ are monic and no term f_i is divisible by $\mathrm{LT}_{\leq}(f_j)$ for any pair $i \neq j$, where $\leq$ is monomial order.*

2.6 Groebner fans

Groebner bases for a fixed Ideal I with different monomial orders can look very different and have different properties. The difference can be in the number of elements in the Groebner basis, the length or the degree of the elements. So it will be helpful if all possible Groebner basis of a fixed ideal can be collected together.
Even if there are infinite monomial orders for an ideal, the amount of reduces Groeber bases are finite. [4]

Definition 2.10 (Groebner fan) *[4] A Groebner fan of an Ideal I consist of finitley many closed convex polyedral cones with vertices at the origin, such that*

- *A face of a cone σ is $\sigma \cap \{l = 0\}$, where $l = 0$ is a nontrivial linear equation such that $l \geq 0$ on σ. Any face of a cone in the fan is also in the fan.*
- *The intersection of two cones in the fan is a face of each.*

In order to construct a Groebner fan to a given Ideal, consider the marked Groebner basis $G = \{g_1, \cdots, g_t\}$ of the Ideal I. A marked Groebner basis is a Groebner basis where each $g \in G$ has an identified leading term, such that G

is a reduced Groebner basis with respect to some monomial order $>$ selecting those terms. Informally, where all leading terms in G are marked.

The elements of G, g_i , can be written as

$$g_i = x^{\alpha(i)} + \sum_{\beta} c_{i,\beta} \cdot x^{\beta},$$

where $x^{\alpha(i)}$ is the leading term and $x^{\alpha(i)} > x^{\beta}$,with respect to a monomial order, whenever $c_{i,\beta} \neq 0$.
Now if a weight vector $\mathbf{w}$ fullfills the inequation $\alpha(i) * \mathbf{w} \geq \beta * \mathbf{w}$, the vector selects the correct leading term in g_i as the term with the highest weight.

So the cone of a Groebner basis can be written as [4]

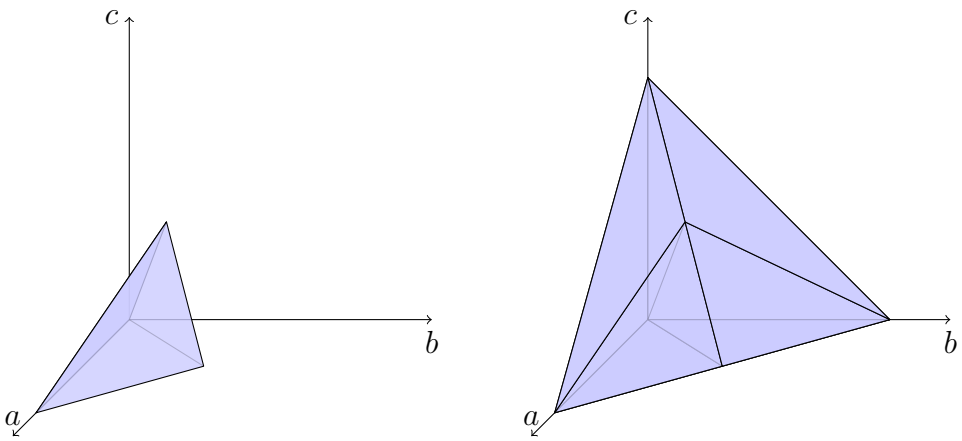
$$C_G = \left\{ \mathbf{w} \in (\mathbb{R}^n)^+ : \alpha(i) \cdot \mathbf{w} \geq \beta \cdot \mathbf{w} \quad \text{whenever } c_{i,\beta} \neq 0 \right\}$$

Example 6 Consider the Ideal with $I = \langle x^2 - z, y - x \rangle \in \mathbb{Q}[x, y, z]$. Note that the ring is 3-dimensional so that Groebner fan can be plotted in the positive orthant $\mathbb{R}^3_+$.
The marked Groebner basis with respect to the *lex* order with $x > y > z$ is

$$G^{(1)} = \left\{ \underline{y^2} - z, \underline{x} - y \right\}$$

The leading terms are underlined. Let $\mathbf{w} = (a, b, c) \in \mathbb{R}^3_+$. Then $\mathbf{w}$ is in the cone C_{G^1} if and only if the inequalities defined above are satisfied.

- $(2, 0, 0) \cdot (a, b, c) \geq (0, 0, 1) \cdot (a, b, c)$ or $2a \geq c$
- $(0, 1, 0) \cdot (a, b, c) \geq (0, 0, 1) \cdot (a, b, c)$ or $b \geq a$



(a) Groebner fan for the first basis (b) Complete Groebner fan

Figure 1: Groebner fans for the given example

This is the first cone of the Groebner fan and can be drawn in the positive orthant sliced of the plane of $a + b + c = d$, $d \in \mathbb{R}$ for visuality.

This figure shows that the Groebner fan is not complete, since the cone does not cover the whole positive orthant. In this example, the other reduced Groebner basis can be obtained by applying the Buchberger Algorithm with common term orders to the Ideal I. If the computed cones still are not the the whole positive orthant, then a further computation with weight vectors are necessary.

$\Diamond$

The example illustrates clearly that a arbitrary non-negative weight vector can be selected and if the vector lies in a certain cone, the corresponding Groebner base will match with respect to the weight vector.

This is strategy is reasonable for a small example like above. In genearal, the whole Groebner fan can be computed with the Groebner walk. See [4] for further details.

An inexpensive way to obtain all reduced Groebner bases of a special ideal, the Code Ideal, which will be explained in section 2.8.

2.7 Toric Ideals

This work and is focused on Code Ideals, so it is useful to define the toric Ideals first. Given a matrix $A = [a_1, \ldots, a_n] \in \mathbb{Z}^{d \times n}$ and $u \in \mathbb{Z}^n$, which can be decomposed in u^+ and u^-, where u^+ and u^- have nonnegative coefficients and disjoint support.

Definition 2.11 (Toric Ideal) *[5] A toric ideal I_A is defined as*

$$\boldsymbol{I}_A = \langle \boldsymbol{x}^{u^+} - \boldsymbol{x}^{u^-} \mid u \in ker\,(A) \rangle$$

The toric ideal can also be expressed as

$$\mathbf{I}_A = \langle \mathbf{x}^u - \mathbf{x}^u \mid Au = Av, u, v \in \mathbb{N}_0^n \rangle.$$

2.8 Enumerating Groebner fans

In this section, 2 algorithms will be explained with the purpose to enumerate the Groebner fan. To compute all Groebner bases from a toric ideal I_A , it is necessary to search the *edge graph* of a Groebner fan.
Two reduced Groebner bases with respect to a term order covered by the generic weight vectors c_1,c_2 are said to be adjacent if the two Groebner cones share a common *facet*.

Definition 2.12 (Facet Binomial) *[3] The binomial $x^{\alpha_k} - x^{\beta_k} \in \mathcal{G}_c$ is a facet binomial of $\mathcal{G}_c$ if and only if there exists a vector $u \in \mathbb{R}^n$ which satisfies :*

- $\{\alpha_i \cdot u > \beta_i \cdot u : i = 1, \cdots, t, i \neq k\}$
- $\{\beta_k \cdot u > \alpha_k \cdot u\}$

Computing the facet binomials of a reduced Groebner basis $\mathcal{G}$ can be computationally expensive, because it is needed to solve as many linear programs as the cardinality of $\mathcal{G}$. The algorithm for finding the facets can be as follows.

Algorithm 3 Finding the facets of a reduced Gröbner bases of I_A [3]

Input: Reduced Gröbner basis $\mathcal{G} = \{\underline{x}^{a_i} - x^{b_i} : i = 1, \ldots, t\}$ of I_A
Output: The facet binomials of $\mathcal{G}$

```
1: Facets := ∅
2: for each binomial x^{a_i} − x^{b_i} in G do
3:     if a_i − b_i ∉ {a_j − b_j : x^{a_j} − x^{b_j} ∈ G} then
4:         Facets := Facets ∪ {x^{a_i} − x^{b_i}}
5:     end if
6: end for
7: return Facets
```

Example 7 Given the reduced Groebner base $\mathcal{G}$

$\Diamond$

An other way to find the facets without linear programming is possible with the property of the following initial ideals.

Let $\mathcal{G}_c$ be a reduced Groebner basis with respect to a generic weight vector c. Then $x^a - x^b$ is a facet binomial of $\mathcal{G}_c$ only if $in_c(I_a)$ is the initial Ideal of $W_{a-b} = \langle x^a - x^b \rangle + \langle x^c : x^c \text{ is a minimal generator of } in_c(I_a), x^c \neq x^a \rangle$.[3] With the help of this property it is possible to define an algorithm to find a superset of facet binomials of a reduced Groebner basis.

Algorithm 4 Finding a superset of the facet binomials of a reduced Groebner basis of I_A [3]

Input: Reduced Gröbner basis $\mathcal{G}_c$ of I_A
Output: A superset SS of the facet binomial $\mathcal{G}_c$

```
1: SS := ∅
2: for each x^a − x^b ∈ G_c do
3:     W_{a−b} := ⟨x^a − x^b⟩ + ⟨x^c⟩          ▷ x^c is defined as before
4:     if in_c(I_A) is the initial ideal of W_{a−b} with respect to x_a > x_b then
5:         SS := SS ∪ {x^a − x^b}
6:     end if
7: end for
8: return SS
```

In the example (with the Gröebner fan) the facet binomials of a Gröbner cone determine the border to an other Gröber cone. In order to traverse from a Gröber base $\mathcal{G}_c$ to a neighboured Gröbner base $\mathcal{G}_{c'}$ with the certain facet $x^{\alpha} - x^{\beta}$, a procedure making a local change from $\mathcal{G}_c$ to $\mathcal{G}_{c'}$ is required. This procedure is called flip.

Algorithm 5 Local change of reduced Gröbner bases in I_A [3]

Input: Reduced Gröbner basis $\mathcal{G}$ of I_A
A prescribed facet binomial $\underline{x}_i^a - x_i^b \in \mathcal{G}$

Output: The reduced Gröbner basis is adjacent to $\mathcal{G}$ in which $\underline{x}_i^b - x_i^a$ is a facet binomial.

1: Old := $\{\underline{x}_i^a - x_i^b\} \cup \{\underline{x}_j^a : \underline{x}_j^a - x_j^b \in \mathcal{G}, j \neq i\}$ ▷ Prescribed Facet and leading terms only in Old
2: Temp := $\{\underline{x}_i^b - x_i^a\} \cup \{\underline{x}_j^a : x_j^a \in Old\}$ ▷ Flipping the facet Binomial
3: New := Reduced Gröbner basis with respect to the new marking ▷ See algorithm 2.
4: $\mathcal{G}' = \left\{\underline{x}_i^b - x_i^a\right\}$
5: **for each** monomial h in New **do**
6: Reduce h with $\mathcal{G}$ to obtain the monomial h'.
7: Add $h - h'$ to $\mathcal{G}'$ with h marked as the leading term.
8: **end for**
9: Auto-reduce $\mathcal{G}'$ to get $\mathcal{G}_{new}$ ▷ no term shall be divisible by a leading term

This algorithm is correct and can terminate.[3] The main advantage of the algorithm is that no weight vectors must be stored or computed. Weight vectors are carried implicitly and that is possible due to the binomial structure that this subroutine generates for every Gröbner basis.

Example 8 Consider the reduced Groebner basis with respect to a term order $\succ$: $\mathcal{G} = \{x_6^2-1, x_5^2-1, x_4x_5x_6-x_2, x_4^2-1, x_3-x_5, x_2x_6-x_4x_5, x_2x_5-x_4x_6, x_2x_4-x_5x_6, x_2^2-1, x_1-x_5\}$.

Applying the procedure flip$(x_3 - x_5, \mathcal{G})$ leads to:

Old := $\{x_3 - x_5, x_6^2, x_5^2, x_4x_5x_6, x_4^2, x_2x_6, x_2x_5, x_2x_4, x_2^2, x_1\}$
Temp := $\{x_5 - x_3, x_6^2, x_5^2, x_4x_5x_6, x_4^2, x_2x_6, x_2x_5, x_2x_4, x_2^2, x_1\}$

Now the new Groebner basis has to be calculated, but first it useful to know that all pairs of binomials which has the least common multiple of 1 will be auto-reduced to zero. In other words, it is only necessary to form the S-Pairs of binomials that are not relatively prime.

$$\begin{aligned} S(x_5 - x_3, x_5^2) &= x_3 x_5 \\ S(x_5 - x_3, x_4 x_5 x_6) &= x_3 x_4 x_5 \\ S(x_5 - x_3, x_2 x_5) &= x_2 x_3 \\ S(x_5 - x_3, x_3 x_5) &= x_3^2 \end{aligned}$$

Now the new Groebner basis $\mathcal{G}'$ shall be filled with all monomials from $\mathcal{G}$. The underlined terms are the terms reduced with $\mathcal{G}$ and will be the new non-leading terms.

$$\begin{aligned} x_3 x_5 &= x_5(x_3 - x_5) + 1(x_5^2 - 1) + \underline{1} \\ x_3 x_4 x_6 &= x_4 x_6(x_3 - x_5) + 1(x_4 x_5 x_6 - x_2) + \underline{x_2} \\ x_2 x_3 &= x_2(x_3 - x_5) + 1(x_2 x_5 - x_4 x_6) + \underline{x_4 x_6} \\ x_3^2 &= x_3(x_3 - x_5) + x_5(x_3 x_5) + 1(x_5^2 - 1) + \underline{1} \\ x_1 &= (x_1 - x_5) + (x_5 - x_3) + \underline{x_3} \\ x_2 x_6 &= (x_2 x_6 - x_4 x_5) + x_4(x_5 - x_3) + \underline{x_3 x_4} \\ x_2 x_4 &= (x_2 x_4 - x_5 x_6) + x_6(x_5 - x_3) + \underline{x_3 x_6} \end{aligned}$$

The other monomials of $\mathcal{G}$ are not notated here because they did not get a new non-leading term.

This results to the Groebner base :
$\mathcal{G}' = \{x_5 - x_3, x_6^2 - 1, x_5^2 - 1, x_4 x_5 x_6 - x_2, x_4^2 - 1, x_2 x_6 - x_3 x_4, x_2 x_5 - x_4 x_6, x_2 x_4 - x_3 x_6, x_2^2 - 1, x_1 - x_3, x_3 x_5 - 1, x_3 x_4 x_6 - x_2, x_2 x_3 - x_4 x_6, x_3^2 - 1\}$

This Groebner base is not reduced yet. After cancelling all binomials whose terms are divisible by some other leading terms the new reduced Groebner basis is $\mathcal{G}_{new} = \{x_5 - x_3, x_6^2 - 1, x_4^2 - 1, x_2x_6 - x_3x_4, x_2x_4 - x_3x_6, x_2^2 - 1, x_1 - x_3, x_3x_4x_6 - x_2, x_2x_3 - x_4x_6, x_3^2 - 1\}$

$\Diamond$

2.8.1 Breadth first search

In this section an algorithm to enumerate the edge graph of a Groebner fan via breath-first search and its drawbacks are presented.

Algorithm 6 Enumerating the edge graph of the Gröbner fan via breath-first search [3]

Input: Any reduced Gröbner basis $\mathcal{G}_0$ of I_A
Output: All reduced Gröbner bases of I_A, (all vertices of the edge graph)

```
1: Todo := [G_0]
2: Verts := []
3: while Todo ≠ ∅ do
4:     G := first element in (Todo)
5:     Remove G from Todo
6:     add G to Verts
7:     determine list L of facet binomials of G      ▷ With linear programming
8:     for each x^α − x^β ∈ L do
9:         G' = flip(G, x^α − x^β)
10:        if G' ∉ Todo ∪ Verts then
11:            add G' to Todo
12:        end if
13:    end for
14: end whilereturn Verts
```

This algorithm is intuitive but has the drawback that every vertex of the edge graph must be stored and every vertex must be checked against all other vertices if it is a new vertex or not. The more vertices the edge graph has, the more expensive the calculation will be. Also the need of memory will arise if a Gröbner fan has a lot of cones.

2.8.2 Reverse Search tree

The disadvantages can be canceled out with a memoryless algorithm, which runs linear depending on the size of output. The idea is to enumerate the edge graph with depth-first reverse search. The result will be a directed subgraph of the edge graph, called reverse search tree $T_{\succ}(I_A)$.

Definition 2.13 (Mismarked Polynomial) *[3] A polynomial f that has been marked with respect to the monomial order $>$ is mismarked with respect to the monomial order $>'$.*

Example 9 Consider the reduced Gröbner base
$\mathcal{G} = \{x^2y - z, y^2 - xz, zy - xy^2z\}$ with a certain monomial order $>$, which is not the lexicograpic order. Then, the second and last term are clearly mismarked with respect to $>_{lex}$.
Applying the flip-procedure $(\mathcal{G}, y^2 - xz)$ leads to the Groebner base $\mathcal{G} = \{x^2y - z, xz^2 - y^2, zy - xy^2z\}$. Now only the last binomial is mismarked and using flip($\mathcal{G}$, $zy - xy^2z$) the result is the reduced Groebner basis with respect to the lexicographic order $\mathcal{G} = \{xy^2z - xy, xz^2 - y^2, xy - z\}$. Note that every monomial can not be divided by the leading terms, so all Groebner bases are reduced and no auto-reduce is necessary.

$\Diamond$

The *reverse search tree* $T_{\succ}(I_A)$ with a given monomal order $\succ$ can be defined as follows.

Definition 2.14 (Reverse Search Tree) *[3] For two reduced Groebner bases $\mathcal{G}_i$ and $\mathcal{G}_j$, $[\mathcal{G}_i, \mathcal{G}_j]$ directed from $\mathcal{G}_i$ to $\mathcal{G}_j$ is an edge of $T_{\succ}(I_A)$ if $\mathcal{G}_\rangle$ is obtained from $\mathcal{G}_|$ by the procedure flip($\mathcal{G}_i$,$x^\alpha - x^\beta$). x^α is lexicographically maximal among all facet binomials of $\mathcal{G}_i$, that are mismarked with respect to $\succ$.*

[3] ensures that the reverse search tree is an acyclic graph with a unique sink and from any reduced Groebner basis of $\mathcal{G}_c$ of a toric Ideal I_A, there is a unique path to the sink $\mathcal{G}_\succ$. Unlike the Groebner walk procedure, there are still no weight vectors involved, but in return every facet of the Groebner cone must be computed, which can be computionally expensive for reduced Groebner basis with many polynomials.

Algorithm 7 Enumerating the edge graph of the Gröbner fan via reverse search [3]

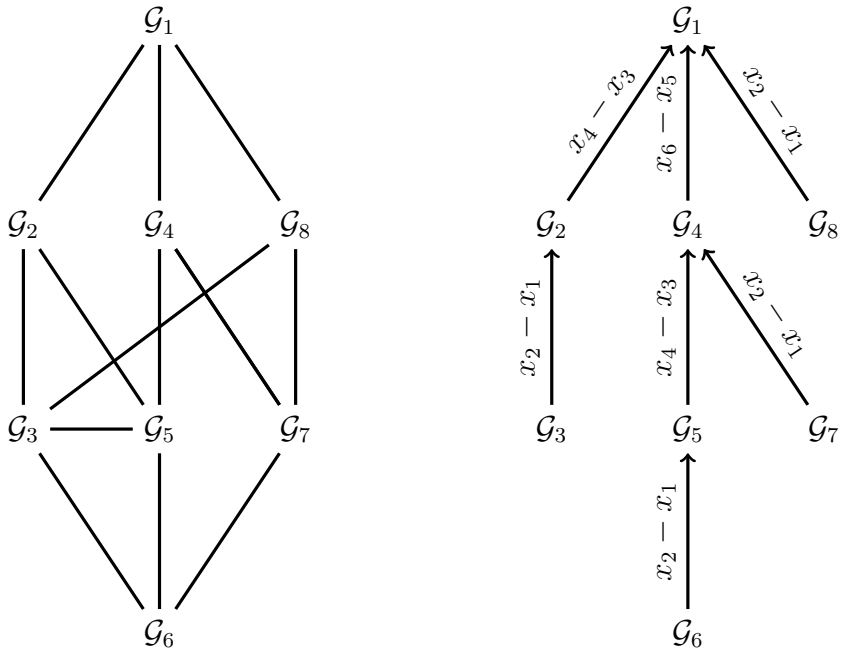
Input: Any reduced Gröbner basis $\mathcal{R}_\succ$ of I_A and its term order $\succ$
Output: All reduced Gröbner bases of I_A, (all vertices of the edge graph)

$\mathcal{G} := \mathcal{R}_\succ$; $j := 0$; $L :=$ list of facet binomials of $\mathcal{G}$ marked by $\succ$
add $\mathcal{G}$ to output
repeat
    while $j < |L|$ **do**
        j := j + 1
        $\mathcal{G}' :=$ flip($\mathcal{G}$,$L[j]$);
        if $[\mathcal{G}', \mathcal{G}] \in T_\succ(I_A)$ **then** ▷ Check for adjacency
            $\mathcal{G} := \mathcal{G}'$; $j := 0$
            $L :=$ list of facet of $\mathcal{G}$ marked by $\succ$
            add $\mathcal{G}$ to output
        end if
    end while
    if $\mathcal{G} \neq \mathcal{R}_\succ$ **then**
        $\mathcal{G}' :=$ unique element such that $[\mathcal{G}', \mathcal{G}] \in T_\succ(I_A)$
        $j := 0$
        $L :=$ list of facets of $\mathcal{G}'$ marked by $\succ$
        repeat
            $j := j + 1$
        until the common facet of $\mathcal{G}$ and $\mathcal{G}'$ is the j-th facet of L
    end if
until $\mathcal{G} = \mathcal{R}_\succ$ and $j = |L|$

Example 10 Consider the Ideal with the reduced Groebner basis with respect to the lexicographic order $x_1 > \ldots > x_6$

$$\mathcal{G} = \{x_1 - x_2, x_3 - x_4, x_5 - x_6, x_2^2 - 1, x_4^2 - 1, x_6^2 - 1\}$$

Applying the breath-first search algorithm for this reduced Groebner basis results to the following edge graph.
Using the reverse search method, this search tree on figure 2b results.



(a) Complete edge graph (b) Reverse search tree

Figure 2: Comparison between the breath-first search and the reverse search

The flipping binomials are written on the edges.

The Groebner bases are:

$$\begin{aligned}
\mathcal{G}_1 &= \{x_1 - x_2, x_3 - x_4, x_5 - x_6, x_2^2 - 1, x_4^2 - 1, x_6^2 - 1\} \\
\mathcal{G}_2 &= \{x_1 - x_2, x_4 - x_3, x_5 - x_6, x_2^2 - 1, x_3^2 - 1, x_6^2 - 1\} \\
\mathcal{G}_3 &= \{x_2 - x_1, x_4 - x_3, x_5 - x_6, x_1^2 - 1, x_3^2 - 1, x_6^2 - 1\} \\
\mathcal{G}_4 &= \{x_1 - x_2, x_3 - x_4, x_6 - x_5, x_2^2 - 1, x_4^2 - 1, x_5^2 - 1\} \\
\mathcal{G}_5 &= \{x_1 - x_2, x_4 - x_3, x_6 - x_5, x_2^2 - 1, x_3^2 - 1, x_5^2 - 1\} \\
\mathcal{G}_6 &= \{x_2 - x_1, x_4 - x_3, x_6 - x_5, x_1^2 - 1, x_3^2 - 1, x_5^2 - 1\} \\
\mathcal{G}_7 &= \{x_2 - x_1, x_3 - x_4, x_6 - x_5, x_1^2 - 1, x_4^2 - 1, x_5^2 - 1\} \\
\mathcal{G}_8 &= \{x_2 - x_1, x_3 - x_4, x_5 - x_6, x_1^2 - 1, x_4^2 - 1, x_6^2 - 1\}
\end{aligned}$$

Even for this small example a lot of edges were saved.

$\Diamond$

2.9 Degree compatible Groebner

Computing the whole Groebner fan can be very expensive and not every Groebner base is interesting. In this section, the degree compatible Groebner fan is introduced and how the algorithm can be changed so that only the degree compatible Groebner fan will be computed.

Definition 2.15 (Degree compatible Groebner basis) *[6] A reduced Groebner basis for an ideal I with respect to a certain monomial order is degree compatible if and only if the corresponding Groebner cone contains the all-one vector 1.*

Equivalent to this, the leading term of every polynomial must have the highest degree. Since a Groebner fan is homogeneous at $\mathbb{R}^n_+$, there will be at least one degree compatible Groebner basis. That is a special case can be easily determined as follows.

Definition 2.16 (Only degree compatible Groebner basis) *[6] A Groebner basis $\mathcal{G}$ with respect to a degree compatible monomial ordering $>$ is the only degree compatible Groebner basis for an Ideal if and only if*

$$deg(x^a) > deg(x^b) \quad \forall \quad x^a - x^b \in \mathcal{G}$$

This can be also described by the all-one vector that lies completely in a Groebner cone of a Groebner basis $\mathcal{G}$. It follows that the all-one vector does not intersect with any facets if there is only one degree compatible Groebner basis.

The algorithms 4 and 5 can be adapted in order to compute only the degree compatible Groebner fan.
The breadth-first search now needs a degree compatible Groebner basis as an input. This can be achieved by applying the Buchberger Algorithm with a degree compatible monomial, for example the *grlex* order. After that it is required that the Groebner basis is checked if its is the only degree compatible basis. Also the only facet binomials x^a - x^b which are allow to be "flipped" are the binomials that fulfills the condition $deg(x^a) = deg(x^b)$.

The reverse search tree can be deployed as in definition 2.14 but with the restriction that $deg(x^a) = deg(x^b)$ must be fulfilled to traverse the degree compatible Groebner fan. Lemma 2.2 from [6] guarantees that at least one such facet binomial will be found. The sink of the reverse search tree contains binomials that are not mismarked with respect to some degree compatible monomial order.

2.10 Linear Codes over Prime Fields

This focus of this work is computing the Groebner fan of linear codes. Now the mathematic background of the Groebner fans is given, the linear Codes and Code Ideals have to be defined to give a connection between these two topics.
Let $\mathbb{F}$ be a finite field and let n and $k \in \mathbb{N}$ with $n \geq k$.

Definition 2.17 (Linear Code) *[5] A linear code of length n and dimension k over $\mathbb{F}$ is the image $\mathcal{C}$ of a injective linear mapping $\phi : \mathbb{F}^k \to \mathbb{F}^n$*

Such a code will be denoted als an [n,k] code and its elements are called codewords. The codewords are written as row vectors. The Code $\mathcal{C}$ can alternatively be described as row space matrix of $G \in \mathbb{F}^{k \times n}$. The rows of G form a basis of $\mathcal{C}$. G is also calles *generator matrix* for $\mathcal{C}$.

Definition 2.18 (Standard form) *[5] A $[n, k]$ code $\mathcal{C}$ is in standard form if it has a generator matrix like $G = (I_k|M)$, where I_k is the $k \times k$ matrix.*

Example 11 Consider the binary $[7, 4]$ Hamming Code with its generator matrix

$$G = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 \end{pmatrix}$$

The code c of the word x is obtained with the vector-multiplication

$$xG = c$$

Let x be $(1,0,1,0)$, then the codeword c results to:

$$(1,0,1,0)\cdot\begin{pmatrix}1&0&0&0&1&1&0\\0&1&0&0&1&0&1\\0&0&1&0&1&1&1\\0&0&0&1&0&1&1\end{pmatrix}=(1,0,1,0,0,0,1)$$

$\Diamond$

Two codes are equivalent if one generator matrix can be obtained from the other by permutating columns and rows. It follows that every linear code is equivalent to a linear code in standard form.[5]

A linear Code $\mathcal{C}$ can be *punctured* by deleting the same coordinate i in each codeword.

2.11 Code Ideals

In this section, the linear codes and the Groebner bases come together.
Each linear code $\mathcal{C}$ can be associated to a binomial ideal.[*dueckpaper*] Let $\mathcal{C}$ be a $[n,k]$ code and let $\mathbb{K}[\mathbf{x}] = \mathbb{K}[x_1,\cdots,x_n]$. Then the *code ideal* can be defined as follows:

Definition 2.19 (Code Ideal) *[6] A code ideal $I(\mathcal{C})$ is the union between the toric ideal and a nonprime ideal I_p, such that*

$$I_{\mathcal{C}} = \langle \boldsymbol{x}^{c} - \boldsymbol{x}^{c'} | c - c' \in \mathcal{C}\rangle + I_p,$$
$$\textit{where } I_p = \langle x_i^p - 1 | 1 \leq i \leq n\rangle$$

Example Let $\mathcal{C}_1$ be a binary $[6,3]$ code with generator matrix

$$G_1 = \begin{pmatrix}1&0&0&0&1&0\\0&1&0&1&1&1\\0&0&1&0&1&0\end{pmatrix}$$

The associated code Ideal $I(\mathcal{C})$ leads to:

$$\begin{aligned} I(\mathcal{C}) =& \{x_1 - x_5,\ x_2 - x_4x_5x_6,\ x_3 - x_5\} \cup \\ & \{x_1^2 - 1,\ x_2^2 - 1,\ x_3^2 - 1,\ x_4^2 - 1,\ x_5^2 - 1,\ x_6^2 - 1\} \end{aligned}$$

Note that the terms $x_1^2-1,\ x_2^2-1,\ x_3^2-1$ are divisible by the leading terms of the toric ideal. The reduced Groebner basis $\mathcal{G}_>$ with respect to the lexicographic ordering $>$ with $x_1 > \cdots > x_6$ is:

$$\mathcal{G}_\succ = \{x_1 - x_5,\ x_2 - x_4x_5x_6,\ x_3 - x_5\} \cup \{x_4^2 - 1,\ x_5^2 - 1,\ x_6^2 - 1\}$$

$\Diamond$

3 Software

This section is all about the practical part of this work. At first, a brief reason why the software is implemented in C is given. Secondly, an accurate description is presented of how the software can be compiled and used for own demands. After that, the software is tested on some randomly generated linear Codes. The number of degree compatible and all Groebner bases are presented and comparision of the operational time agains Gfan[7]is presented.
This software is a re-implementation of TiGERS [3]. All features that are needed for the code ideals were added, also the adapted algorithms for computing degree compatible Groebner bases with reverse search and breath-first search were implemented. Additional features are explained in Section 3.2. The programming language of choice is C because it is fast and the capability to make own data structures are simple and sufficient enough.

3.1 Data Structures

With the special attribute that code ideals only contains binomials and monomials and reduced Groebner bases have always the coefficent 1, only the exponent vectors representing the monomials have to be stored.

Listing 1: Data structure of binomials [3]

```
typedef struct bin_tag *binomial;
struct bin_tag{
    int *exps1;
    int *exps2;
    int *E;
    int ff;
    int bf;
    binomial next;
};
```

The pointer *exps1* stores the exponent vector of the first monomial and *exps2* does it with the second monomial. The integer *ff* is a flag which shows if a

binomial is a facet binomial or not and *bf* tells if there is a monomial or binomial. The pointer binomial next indicates that binomials are linked together like a linear list, which is necessary to describe ideals and Groebner bases.

The next code snippet shows the other important data structure. Again, it is a linked list like the binomials with the purpose to connect all reduced Groebner together, which is needed for the breath-first search.

The first four integers show off the identification number of the vertex of the edge graph, the number of facet binomials, number of binomials and the highest degree. Next is a pointer to the next generating set of the linked list.

Listing 2: Data structure of generating sets

```
typedef struct gset_tag *gset;

/* Linked List of gset_tag which contiains the binomial and the
   caching informations*/
struct gset_tag{
    int id;
    int nfacets;
    int nelts;
    int deg;
    binomial bottom;
    binomial cache_edge;
    struct gset_tag *cache_vtx;
    struct gset_tag *next;
};
```

3.2 Manual

This software was programmed and evaluated with Linux, so at first, it is needed to compile the software. The makefile is given and it only takes the console, moving to the direction where folder is and typing 'make'.
All inputs, outputs, options and flags are passed with the commandline argu-

ments. At first it is useful to run the program with the purpose to print the help-message only with the command `./cidgel -h`.

Listing 3: Code Snippet of the help-message

```
static char *helpmsg[] = {
    "Function: Enumerate all or d.c Groebner bases of a code ideal I(C).",
    " \n",
    "Options:\n",
    " -h print this message\n"
    " -i (filename) set file name for input [default: stdin]\n",
    " -o (filename) set file name for output [default: stdout]\n",
    " -m (filename) set file name for code-matching \n",
    " -R only compute root of tree \n",
    " -r compute all grobner bases [done by default]\n",
    " -C turn partial caching on [done by default]\n",
    " -c turn partial caching off \n",
    " -T print edges of search tree \n",
    " -t do not print edges of search tree [assumed by default]\n",
    " -L print vertices by giving initial ideals\n",
    " and printing facet biomials.\n",
    " -l print vertices as grobner bases [done by default]\n",
    " -F Use only linear algebra when testing facets [default]\n",
    " -f use FLIPPABILITY test first when determining facets\n",
    " -e use exhaustive search instead of reverse search\n",
    " -E use reverse search [default]\n",
    " -d degree compatible Groebner bases only \n",
    " -n do not print vertices or edges \n",
    " -p calculate Groebner fans of punctured codes \n",
    NULL
};
```

The listing above shows all options that are available. These flags can be passed in arbitrary order to the program. It is necessary to write a matrix and storing it into a file to give the program an input.

For example, the data has the name '`matrix`', has the following content and is in the same folder as the compiled software.

Listing 4: Example-input

```
M: { 6 10 2 :
1 0 0 0 0 0 0 0 1 1
0 1 0 0 0 0 1 1 0 0
0 0 1 0 0 0 1 1 1 1
0 0 0 1 0 0 0 1 1 1
0 0 0 0 1 0 1 0 0 0
0 0 0 0 0 1 0 1 0 1
}
```

This Matrix is a generator matrix for a binary $[10, 6]$ code in essential standard form. The first number in the first row gives the code dimension, the second tells the length of the codeword and the last number indicates in which primary field the generator matrix shall be evaluated.

Now if the degree compatible Groebner bases of this generator matrix without printing the Groebner basis shall be written in a outputfile called `Example-output`. Additionally the linear programming shall be leaved out, then the program call is: `./cidgel -i Example-input -o Example-output -d -n -f`. The user do not has to specify an output file, the result will be printed in the console then.

Listing 5: Snippet of the example output

```
R: 10
G: {a−i∗j, b−g∗h, c−g∗h∗i∗j, d−h∗i∗j, e−g, f−h∗j, gˆ2−1, hˆ2−1, iˆ2−1, jˆ2−1}

 Enumerating degree compatible Groebner bases
    using reverse search
    taking input from randomgenerator/10_6
    with partial caching
    using wall ideal pretest for facet checking

Number of Groebner bases found 216
Number of edges of state polytope 792
max caching depth 10
max facet binomials 18
min facet binomials 12
max binomials in GB 41
min binomials in GB 40
max degree 3
min degree 2
randomgenerator/10_6: Reverse Search, Caching, A−pretest,
time used (in seconds) 42.16
```

3.3 Computational experience

In this section the difference between degree compatible Groebner bases and all Groebner bases from a linear Code are researched. Furthermore, the time for all Grooebner bases is compared against Gfan.

3.4 Documentation and electronic availability

A HTML-based documentation of the software is created with the help of doxygen.[*doxygen*]

4 Future Work

5 Conclusion

References

[1] Mehwish Saleemi. *Coding Theory via Groebner Bases*. PhD thesis, Technischen Universitat Hamburg-Harburg, 2012.

[2] Karl-Heinz Zimmermann. *Algebraic Statistics*. July 2009.

[3] Rekha R. Thomas Birkett Huber. Computing gröbner fans of toric ideals. Technical report, Institute for Defense Analysis, United States, Department of Mathematic, University of Washington, 1999.

[4] David A. Cox, John Little, and Donal O'Shea. *Ideals, Varieties, and Algorithms: An Introduction to Computational Algebraic Geometry and Commutative Algebra, 3/e (Undergraduate Texts in Mathematics)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2007.

[5] Karl-Heinz Zimmermann Natalia Dück. Universal gröbner bases for binary linear codes. *International Journal of Pure and Applied Mathematics*, 2013.

[6] Natalia Dück. Computation of the d.c. gröbner fan. April 2014.

[7] Anders Nedergaard Jensen. The gfan homepage, April 2005.