On Signature-based Gröbner Bases over Euclidean Rings

Christian Eder Gerhard Pfister Adrian Popescu

Department of Mathematics, University of Kaiserslautern
Erwin-Schrödinger-Str.,
67663 Kaiserslautern, Germany
{ederc,pfister,popescu}@mathematik.uni-kl.de

ABSTRACT

In this paper we present first steps in using signature-based Gröbner basis algorithms like Faugère's F5 or GVW for computation over Euclidean rings. We present problems appearing when having to deal with coefficients and zero divisors and give practical solution techniques. A hybrid algorithm is presented trying to combine the advantages of signature-based and non-signature-based Gröbner basis computation. For some examples speedups are achieved due to faster finding good reducers with the hybrid technique.

Keywords

Gröbner bases, Signature-based algorithms, Euclidean rings

1. INTRODUCTION

In 1965 [1, 4], Buchberger initiated the theory of Gröbner bases by which many fundamental problems in mathematics, science and engineering can be solved algorithmically. Specifically he introduced some key structural theory, and based on this theory, proposed the first algorithm for computing Gröbner bases. Buchberger's algorithm introduced the concept of critical pairs and repeatedly carries out a certain polynomial operation (called reduction).

Many of those reductions would be determined as "useless" (i.e. no contribution to the output of the algorithm), but only a posteriori, that is, after an (often expensive) reduction process. Thus intensive research was carried out, starting with Buchberger, to avoid the useless reductions via a priori criteria, see, for example, [2, 3, 11].

In [9] Faugère presented the F5 algorithm which uses signatures in order to detected redundant computations in advance. For regular sequences as input F5 does not compute any zero reduction at all. Over the years many new variants of signature-based Gröbner basis algorithms have been presented, we refer to [6] for a survey on this class of algorithms. All these algorithms have in common that they compute Gröbner bases in polynomial rings over fields.

Assuming the integers as ground ring, Gröbner basis computations change in the way that one wants to compute a so-called strong Gröbner bases. Such a basis is achieved by not only handling usual critical pairs consisting of S-polynomials, but also so-called GCD-polynomials (see for example [14, 13]). In 1988, Kandri-Rody and Kapur gave first algorithms for computing Gröbner bases over Euclidean domains [12].

*ISSAC '17, July *-*, 2017, Kaiserslautern, Germany*
© 2017 ACM. ISBN 978-1-4503-4380-0/16/07…$15.00
DOI: http://dx.doi.org/10.1145/2930889.2930914

Here we generalize signature-based Gröbner basis algorithms to computation over Euclidean rings, in particular, the integers. In Section 2 we introduce basics and notation. Sections 3 – 6 discuss problems and possible solutions when computing over Euclidean rings. In particular, we present the problem of signature drops and how to keep them at a minimum. Moreover, techniques for improving coefficient growth and other computational overhead are given. In Section 7 we show how signature-based computation can be efficiently used as a prereduction step for a classical Gröbner basis computation over Euclidean rings and give experimental results in Section 8. Section 9 covers problems over rings with zero divisors.

2. BASIC NOTATIONS

We use notation corresponding to [6]: Let $\mathscr{R}$ be an Euclidean ring. Besides Section 9 we assume that $\mathscr{R}$ has no zero divisors. A *polynomial* $f = \sum_v a_v x^v := \sum_{v\in\mathbb{N}^n}^{\text{finite}} a_{v_1,\ldots,v_n} \prod_{i=1}^n x_i^{v_i}$ in n variables $x_1,\ldots,x_n$ over $\mathscr{R}$ is a finite $\mathscr{R}$-linear combination of *terms* $a_{v_1,\ldots,v_n} \prod_{i=1}^n x_i^{v_i}$ such that $v \in \mathbb{N}^n$ and $a_v \in \mathscr{R}$. The *polynomial ring* $\mathscr{P} := \mathscr{R}[x] := \mathscr{R}[x_1,\ldots,x_n]$ in n variables over $\mathscr{R}$ is the set of all polynomials over $\mathscr{R}$ together with the usual addition and multiplication. For $f = \sum_v a_v x^v \neq 0 \in \mathscr{P}$ we define *the degree of* f by $\deg(f) := \max\{v_1 + \cdots + v_n \mid a_v \neq 0\}$. For $f = 0$ we set $\deg(f) := -1$. For $m > 0$ we denote by $\mathscr{P}^m$ a *free* $\mathscr{R}$*-module* and let $e_1,\ldots,e_m$ be the standard basis (unit vectors) in $\mathscr{P}^m$. A *module element* $\alpha \in \mathscr{P}^m$ can be written as a finite sum $\alpha = \sum_{a\in\mathscr{P}}^{\text{finite}} a e_i$. The elements ae_i are the *module terms* of α. A *module monomial* is an element of $\mathscr{P}^m$ with exactly one term. A module monomial with coefficient 1 is *monic*. Neither module monomials nor terms of module elements are necessarily monic. Let $\alpha \simeq \beta$ for $\alpha,\beta \in \mathscr{P}^m$ if $\alpha = a\beta$ for some non-zero $a \in \mathscr{R}$.

Let $(f_1,\ldots,f_m) \in \mathscr{P}$ be a finite sequence of polynomials. We define a module homomorphism $\pi : \mathscr{P}^m \to \mathscr{P}$ by $e_i \mapsto f_i$ for all $1 \leqslant i \leqslant m$. We use the shorthand notation $\overline{\alpha} := \pi(\alpha) \in \mathscr{P}$ for $\alpha \in \mathscr{P}^m$. An element $\alpha \in \mathscr{P}^m$ with $\overline{\alpha} = 0$ is called a *syzygy*. The module of all syzygies (of $(f_1,\ldots,f_m)$) is denoted $\operatorname{syz}((f_1,\ldots,f_m))$.

Let $<$ denote two different orders — one for $\mathscr{P}$ and one for $\mathscr{P}^m$:

1. $<$ for $\mathscr{P}$ is a *monomial order*, which is a well-order on the set of monomials in $\mathscr{P}$ such that $a < b$ implies $ca < cb$ for all monomials $a,b,c \in \mathscr{P}$.
2. $<$ for $\mathscr{P}^m$ is a *module monomial order*, which is a well-order on the set of module monomials in $\mathscr{P}^m$ such that $\alpha < \beta$ implies $c\alpha < c\beta$ for all module monomials $\alpha,\beta \in \mathscr{P}^m$ and monomials $c \in \mathscr{P}$.

CONVENTION 1. *In this paper we restrict ourselves to the* position-over-term order *on $\mathscr{P}^m$: Let $<$ be a monomial order on $\mathscr{P}$ and let ae_i, be_j be two module monomials in $\mathscr{P}^m$. Then we define $<_{pot}$ via*

$$ae_i <_{pot} be_j \text{ if and only if either } i < j \text{ or } i = j \text{ and } a < b.$$

Moreover, if not otherwise noted, we assume $<$ to denote the degree reverse lexicographical order *on $\mathscr{P}$.*

Given such a (module) monomial order $<$ we can highlight the maximal terms of elements in $\mathscr{P}$ or $\mathscr{P}^m$ w.r.t. $<$: For $f \in \mathscr{P} \setminus \{0\}$, $\mathrm{lt}(f)$ is the *lead term*, $\mathrm{lm}(f)$ the *lead monomial*, and $\mathrm{lc}(f)$ the *lead coefficient* of f. For any set $F \subset \mathscr{P}$ we define the *lead ideal* $L(F) = \langle \mathrm{lt}(f) \mid f \in F \rangle$; for an ideal $I \subset \mathscr{P}$, $L(I)$ is defined as the ideal of lead terms of all elements of I. For $\alpha \in \mathscr{P}^m \setminus \{0\}$ we denote the maximal term w.r.t. $<$ by $\mathfrak{s}(\alpha)$, the *signature of* α. We also define signatures of polynomials: For $f \in \mathscr{P}$, $\mathfrak{s}(f)$ is *a* signature of f, given by the signature of the corresponding module element $\alpha \in \mathscr{P}^m$ mapping to f: $\mathfrak{s}(f) = \mathfrak{s}(\overline{\alpha}) = \mathfrak{s}(\alpha)$. Note that whereas $\mathfrak{s}(\alpha)$ is uniquely defined, $\mathfrak{s}(f)$ is not since there exist different module elements with different signatures that map to f.

Our purpose is to compute a Gröbner basis G for a given ideal $I = \langle f_1, \ldots, f_m \rangle \subset \mathscr{P}$. Working over a field there are many equivalent definitions of how to obtain a canonical or normal form when reducing a given polynomial by such a basis G. Working over more general rings these definitions are no longer equivalent and over Euclidean domains, like the integers, this, in particular, results in the term of a strong Gröbner basis we define in the following:

Assuming that our coefficient ring $\mathscr{R}$ is an Euclidean domain we can define a total order $\prec$ using the Euclidean norm of its elements: Let $a_1, a_2 \in \mathscr{R}$, then

$$a_1 \prec a_2 \text{ if } \mid a_1 \mid < \mid a_2 \mid .$$

For example, for the integers we can use the absolute value and break ties via sign: $0 \prec -1 \prec 1 \prec -2 \prec 2 \prec -3 \prec 3 \prec \ldots$

The reduction process of two polynomials f and g in $\mathscr{P}$ depends now on the uniqueness of the minimal remainder in the division algorithm in $\mathscr{R}$:

DEFINITION 1. *We say that g* top-reduces *f if $\mathrm{lm}(g) \mid \mathrm{lm}(f)$ and if there exist $a, b \in \mathscr{R}$ such that $\mathrm{lc}(f) = a\,\mathrm{lc}(g) + b$ such that $a \neq 0$, which coincides with $b \prec \mathrm{lc}(f)$. The top-reduction of f by g is then given by*

$$f - a\frac{\mathrm{lm}(f)}{\mathrm{lm}(g)}g.$$

So top-reduction takes place if the reduced polynomial will have either a smaller lead mononmial or a smaller lead coefficient. Relaxing the reduction of the lead term to any term of f, we say that g reduces *f. In general, we speak of the reduction of a polynomial f w.r.t. a finite set $F \subset \mathscr{P}$.*

The result of such a reduction might not be unique, this is exactly the property Gröbner bases deliver:

DEFINITION 2. *A finite set $G \subset \mathscr{P}$ is called a* Gröbner basis *for an ideal I w.r.t. $<$ if $G \subset I$ and $L(G) = L(I)$. Furthermore, G is called a* strong Gröbner basis[1] *if for any $f \in I \backslash \{0\}$ there exists a $g \in G$ such that $\mathrm{lt}(g) \mid \mathrm{lt}(f)$.*

Clearly, assuming the field case, any Gröbner basis is a strong Gröbner basis. But in our setting with $\mathscr{R}$ being an Euclidean ring one has to check the coefficients, too, as explained in Definition 1.

[1]Note that in the area of signature-based algorithms the notion of a *strong Gröbner basis* is sometimes also used in a different manner, see, for example, [10, 6].

EXAMPLE 1. *Let $\mathscr{R} = \mathbb{Z}/6$ and $I := \langle x \rangle \in \mathscr{R}[x]$. Clearly, $G := \{2x, 3x\}$ is a Gröbner basis for I, but it is not a strong Gröbner basis for I since $2x \nmid x$ and $3x \nmid x$.*

From Example 1 it is clear that using the usual Buchberger algorithm as in the field case will not compute a strong Gröbner basis. Luckily, we can fix this via taking care of two different types of S-polynomials.

DEFINITION 3. *Let $f, g \in \mathscr{P}$. We assume w.l.o.g. that $\mathrm{lc}(f) \prec \mathrm{lc}(g)$. Let $t = \mathrm{lcm}(\mathrm{lm}(f), \mathrm{lm}(g))$, $t_f = \frac{t}{\mathrm{lm}(f)}$, and.$t_g = \frac{t}{\mathrm{lm}(g)}$.*

1. *Let $a = \mathrm{lcm}(\mathrm{lc}(f), \mathrm{lc}(g))$, $a_f = \frac{a}{\mathrm{lc}(g)}$, and $a_g = \frac{a}{\mathrm{lc}(f)}$. The* S-polynomial *of f and g is denoted*
$$\mathrm{spol}(f,g) = a_f t_f f - a_g t_g g.$$
2. *Let $b = \gcd(\mathrm{lc}(f), \mathrm{lc}(g))$. Choose $b_f, b_g \in \mathscr{R}$ such that $b = b_f \mathrm{lc}(f) + b_g \mathrm{lc}(g)$.[2] The* GCD-polynomial *of f and g is denoted*
$$\mathrm{gpol}(f,g) = b_f t_f f + b_g t_g g.$$

REMARK 1. *In the field case $\mathrm{spol}(f,g) = \mathrm{gpol}(f,g)$ (after a possible normalization of the coefficients). Furthermore, if $\mathrm{lc}(f) \mid \mathrm{lc}(g)$ then $b_f = 1$ and $b_g = 0$, thus $\mathrm{gpol}(f,g)$ is just a power product multiple of f.*

Given an ideal $I \subset \mathscr{P}$ a strong Gröbner basis for I can now be computed using Buchberger's algorithm taking care not only of S-polynomials but also GCD-polynomials. We refer, for example, to [13] for more details.

Computing with signature-based Gröbner basis algorithms we have to take care of the signatures. For this we have to redefine the essential structures, i.e. reduction, S- and GCD-polynomials.

DEFINITION 4 ($\mathfrak{s}$-REDUCTION). *Let $\alpha, \beta \in \mathscr{P}^m$. We say that β* top-$\mathfrak{s}$-reduces *α if $\overline{\beta}$ reduces $\overline{\alpha}$ and $\mathfrak{s}(\alpha) > \mathfrak{s}\left(a\frac{\mathrm{lm}(\overline{\alpha})}{\mathrm{lm}(\overline{\beta})}\beta\right)$ where $\mathrm{lc}(\overline{\alpha}) = a\,\mathrm{lc}\left(\overline{\beta}\right) + b$ and $a \neq 0$. As above, we relax this definition to an* $\mathfrak{s}$-reduction *on any term of α. In the same way we speak of the $\mathfrak{s}$-reductionof α w.r.t. to finite set $M \subset \mathscr{P}^m$.*

This concept generalizes to the notion of S-pairs and GCD-pairs.

DEFINITION 5. *Let $\alpha, \beta \in \mathscr{P}^m$. We assume that $\mathrm{lc}(\overline{\alpha}) \prec \mathrm{lc}\left(\overline{\beta}\right)$. Let $t = \mathrm{lcm}\left(\mathrm{lm}(\overline{\alpha}), \mathrm{lm}\left(\overline{\beta}\right)\right)$, $t_\alpha = \frac{t}{\mathrm{lm}(\overline{\alpha})}$, and $t_\beta = \frac{t}{\mathrm{lm}(\overline{\beta})}$.*

1. *Let $a = \mathrm{lcm}\left(\mathrm{lc}(\overline{\alpha}), \mathrm{lc}\left(\overline{\beta}\right)\right)$, $a_\alpha = \frac{a}{\mathrm{lc}(\overline{\beta})}$, and $a_\beta = \frac{a}{\mathrm{lc}(\overline{\alpha})}$. The* S-pair *of α and β is denoted*
$$\mathrm{spair}(\alpha,\beta) = a_\alpha t_\alpha \alpha - a_\beta t_\beta \beta.$$
2. *Let $b = \gcd\left(\mathrm{lc}(\overline{\alpha}), \mathrm{lc}\left(\overline{\beta}\right)\right)$ such that $b = b_\alpha \mathrm{lc}(\overline{\alpha}) + b_\beta \mathrm{lc}\left(\overline{\beta}\right)$. The* GCD-pair *of α and β is denoted*
$$\mathrm{gpair}(\alpha,\beta) = b_\alpha t_\alpha \alpha + b_\beta t_\beta \beta.$$

REMARK 2. *The term S-pair and GCD-pair are used in order to distinguish these elements from S-polynomials and GCD-polynomials for module elements. Note that S-pairs and GCD-pairs get their coefficient and monomial multipliers from the polynomial lead data, not the module lead data. By construction, it follows that $\overline{\mathrm{spair}(\alpha,\beta)} = \mathrm{spol}\left(\overline{\alpha}, \overline{\beta}\right)$ and $\overline{\mathrm{gpair}(\alpha,\beta)} = \mathrm{gpol}\left(\overline{\alpha}, \overline{\beta}\right)$. Note that in the field case $\mathrm{spair}(\alpha,\beta)$ coincides with $\mathrm{gpair}(\alpha,\beta)$ (after a possible normalization of the coefficients).*

[2]Since $\mathscr{R}$ is an Euclidean ring the extended gcd always exists.

Now, in order to compute polynomial Gröbner bases using signatures we can use a slightly modified version of Buchberger's algorithm. Here we state the main differences, and refer to [6] for further details:

1. All initial generators of $I = \langle f_1, \ldots, f_m \rangle$ get a signature, i.e. we define the projection from $\mathscr{P}^m \to \mathscr{P}$ via setting $\overline{e_i} = f_i$ for all $i \in \{1, \ldots, m\}$. With this we fix a signature to the polynomials.
2. We generate S-pairs and GCD-pairs, but store only the polynomial part plus its corresponding signature:

$$\begin{array}{lcll} \text{spair}(\alpha,\beta) & \Longrightarrow & \text{spol}\left(\overline{\alpha},\overline{\beta}\right) & \text{and} \quad \mathfrak{s}(\text{spair}(\alpha,\beta)), \\ \text{gpair}(\alpha,\beta) & \Longrightarrow & \text{gpol}\left(\overline{\alpha},\overline{\beta}\right) & \text{and} \quad \mathfrak{s}(\text{gpair}(\alpha,\beta)). \end{array}$$

3. We choose from the pair set the element of smallest signature w.r.t. the given monomial order $<$.
4. Once an S-polynomial or GCD-polynomial is chosen, we $\mathfrak{s}$-reduce it w.r.t. the intermediate Gröbner basis.

REMARK & CONVENTION 3. *Over fields signature-based algorithms can correctly compute Gröbner bases by chosing any S-pair from the pair set, not necessarily the one with smallest signature, see, for example, Theorem 2.4 in [10]. Choosing new S-pairs by minimal signature ensures that the algorithm handles elements by increasing signature w.r.t. $<$. In the following, we show that this property no longer holds over Euclidean rings when handling S-pairs* and *GCD-pairs, generalizing the concept of $\mathfrak{s}$-reduction(see Definition 4). In order not to mix up out-of-order pair selection and the new results presented in the following, we focus on signature-based algorithms always choosing the next pair by minimal possible signature.*

REMARK 4. *Note that we use notation from [6] and give a clean theoretical description from the module point of view. Still, the algorithms and their implementations are handling polynomial data without the overhead of computations in the module at all. This introduces a limitation to the main steps of the computation: We must ensure not to lose the information stored in the signature since we do not have access to the full module representation and cannot recover it again. Clearly, in a practical implementation we only store the polynomials and their corresponding signatures, i.e. the lead terms of the module representation. In the following we often use the full module representation since it makes proofs easier.*

EXAMPLE 2. *Let $I = \langle f_1, f_2 \rangle \subset \mathbb{Z}[x,y]$ with $f_1 = 5y^2$, $f_2 = 6xy + 4y^2 + 2x$. We illustrate how the first polynomials are added to the Gröbner basis. The first step is to add the polynomials $f_1 = \overline{e_1}$ and $f_2 = \overline{e_2}$ to the intermediate basis G. Next we generate*

$$\begin{array}{rcl} \text{spair}(e_2,e_1) & = & 5ye_2 - 6xe_1, \\ \text{gpair}(e_2,e_1) & = & ye_2 - xe_1. \end{array}$$

Since $\mathfrak{s}(\text{gpair}(e_2,e_1)) = ye_2 < 5ye_2 = \mathfrak{s}(\text{spair}(e_2,e_1))$ we first $\mathfrak{s}$-reduce gpair(e_2,e_1) *and add the new element $f_3 := xy^2 + 4y^3 + 2y$ to G corresponding to $\alpha := ye_2 - xe_1 \in \mathscr{P}^m$.*

Next we generate new pairs: Note that the GCD-polynomials are useless since the lead coefficients are multiples (lc$(\overline{\alpha}) = 1$, *see Remark 1). So we are left with the two S-pairs:*

$$\begin{array}{rcl} \text{spair}(\alpha,e_1) = 5\alpha - xe_1 & = & 5ye_2 - 5xe_1 - xe_1 = 5ye_2 - 6xe_1, \\ \text{spair}(\alpha,e_2) = 6\alpha - ye_2 & = & 6ye_2 - 6xe_1 - ye_2 = 5ye_2 - 6xe_1. \end{array}$$

Note that these two S-pairs correspond to the same element in $\mathscr{P}^m$, thus we have found some ambiguity that we might want to solve using some of the known signature-based criteria we introduce in the next chapter.

The crucial point of the above example is that the signature of spair(α, e_2) is no longer $6ye_2 = \max_{<}\{\mathfrak{s}(6\alpha), \mathfrak{s}(ye_2)\}$. When *constructing* the S-pair we see that the *signature drops* to $5ye_2$.

3. DETECTING USELESS ELEMENTS

Until now signatures are only a small structure we added to Buchberger's algorithm, but we do not exploit them at all. Algorithms like F5 or GVW are well-known for detecting nearly all reductions to zero in advance, see, for example, [9, 10, 6]. There are two signature-based criteria, the *syzygy criterion* and the *rewrite criterion*. Both have the same underlying idea working over fields: When computing a Gröbner basis the algorithm computes new elements by increasing signature and thus reaches at some point a given signature $T \in \mathscr{P}^m$. Assume that $\mathscr{G} \subset \mathscr{P}^m$ denotes the set such that $\{\overline{\alpha} \mid \alpha \in \mathscr{G}\}$ is the intermediate Gröbner basis and $\mathscr{H} \in \mathscr{P}^m$ is the intermediate syzygy module.[3]

LEMMA 5 (SYZYGY AND REWRITE CRITERION). *In a signature-based Gröbner basis computation over a field by increasing signatures we need to handle, for each signature $T \in \mathscr{P}^m$, exactly one $t\alpha \in \mathscr{P}^m$ from the set*

$$\mathfrak{R}_T = \{t\alpha \mid \alpha \in \mathscr{G} \cup \mathscr{H}, t \text{ a term and } \mathfrak{s}(t\alpha) = T\}.$$

PROOF. See, for example, Lemma 6.1 and Lemma 6.2 in [6]. □

So the statement of both, the syzygy and the rewrite criterion, boils down to the following: We have to choose only one such $t\alpha$ for each signature T out of $\mathfrak{R}_T$ to be further reduced. All other elements are useless and can be removed.

1. If $\alpha \in \mathscr{H}$, then $t\alpha$ is a syzygy and we can skip this signature T (syzygy criterion).
2. If $\alpha \in \mathscr{G}$ but $t\alpha$ is not part of any S-polynomial then we can also skip this signature T and go on with the next one (rewrite criterion).

The main idea is now to choose a so-called *rewrite order* on $\mathfrak{R}_T$ and to handle only the minimal element w.r.t. this rewrite order. For more details see [6]. The correctness of this statement relies on the fact that we generate new elements by increasing signature, i.e. there is no drop in the signature during our Gröbner basis computation. Whereas the problem is not essential for the syzygy criterion (we already have $\overline{\alpha} = 0$, so there cannot be any further reductions of $\overline{t\alpha}$), it is crucial for the rewrite criterion: If $t\alpha \in \mathscr{G}$ but not part of an S-pair we only know that $\overline{t\alpha}$ is the "best reduced" element for signature T if there do not appear S-pairs with signatures $< T$ in the upcoming steps of the algorithm. Otherwise we might be able to further reduce $\overline{t\alpha}$ without changing $\mathfrak{s}(t\alpha)$. This is only a minor restriction over fields, but it becomes a problem over Euclidean rings:

Assume a reduction step over a field, $\alpha - t\beta$ for $\alpha, \beta \in \mathscr{P}^m$, $t \in \mathscr{P}$ which was not allowed due to $\mathfrak{s}(t\beta) > \mathfrak{s}(\alpha)$. If α is added to $\mathscr{G}$ and new S-pairs are constructed, also

$$\text{spair}(\alpha,\beta) = t\beta - \alpha$$

is generated. Clearly, $\mathfrak{s}(\text{spair}(\alpha,\beta)) = \mathfrak{s}(t\beta) > \mathfrak{s}(\alpha)$. Sadly this does no longer hold over Euclidean rings:

THEOREM 6. *Computing signature-based Gröbner bases over Euclidean rings we cannot guarantee an order of operations by increasing signatures.*

[3] $\mathscr{H}$ represents the information of zero reductions until this point of the algorithm. Thus, it is dual to $\mathscr{G}$ which represents the non-zero reductions.

PROOF. Assume that we always take the next S-pair by smallest possible signature. We give now an example of the generation of a new S-pair that has a signature smaller than the last element handled by the algorithm:

As above, assume a reduction step $\alpha - ct\beta$ for $\alpha, \beta \in \mathscr{P}^m$, $t \in \mathscr{P}$ a monomial and $c \in \mathscr{R}$, which was not allowed due to $\mathfrak{s}(ct\beta) > \mathfrak{s}(\alpha)$. If α is added to $\mathscr{G}$, in particular $\text{spair}(\alpha, \beta) = ct\beta - \alpha$ is generated. Since $\alpha \in \mathscr{G}$ and the reduction with $ct\beta$ in the previous step did explicitly not take place due to a higher signature, both α and $ct\beta$ are both not detected by the syzygy or the rewrite criterion. So the above S-pair is really constructed. Let $\mathfrak{s}(\alpha) = at_\alpha e_i$ and $\mathfrak{s}(\beta) = bt_\beta e_i$ for $a, b \in \mathscr{R}, t_\alpha, t_\beta \in \mathscr{P}$. If $a \prec bc$ and $t = \frac{t_\alpha}{t_\beta} \in \mathscr{P}$ then $\mathfrak{s}(\alpha) < \mathfrak{s}(ct\beta)$ and $\mathfrak{s}(\text{spair}(\alpha,\beta)) = (cb - a)t_\alpha e_i$. Clearly, it is possible that $cb - a \prec a$, thus $\text{spair}(\alpha, \beta)$ can introduce a new element to $\mathscr{G}$ that has a signature smaller than $\mathfrak{s}(\alpha)$. □

What follows is that we can still apply the syzygy criterion, but no longer the rewrite criterion over Euclidean rings:

COROLLARY 7. *In a signature-based Gröbner basis computation over Euclidean rings we can remove all S-pairs in a given signature T if there exists a syzygy $\gamma \in \mathscr{H}$ such that $\mathfrak{s}(\gamma) \mid T$. Applying Lemma 5 we can no longer guarantee the correctness of the computation.*

PROOF. For a syzygy $\gamma \in \mathscr{H}$ it holds that $\overline{\gamma} = 0$, thus no further reduction can be applied. So the situation from the proof of Theorem 6 cannot arise. □

4. REDUCED INTERMEDIATE BASES

Over fields Eder and Perry have presented in [7] the idea of interreducing intermediate bases when using signature-based algorithms with $<_{\text{pot}}$.

TECHNIQUE 8 (F5C). *Let us assume that we want to compute a Gröbner basis of the ideal $I = \langle f_1, \ldots, f_m \rangle \subset \mathscr{P}$ w.r.t. $<_{pot}$. The algorithm handles elements by increasing module position, so we assume that the algorithm has finished all elements of index i and wants to go on with elements of index $i+1$:*

1. *Usually the signature-based algorithm does not compute a reduced intermediate Gröbner basis G_i up to the given module position i.*
2. *So one can reduce G_i with the drawback of losing the signature information; by construction, reductions that are not $\mathfrak{s}$-reductions are done in this step. We end up with $G_i = \{g_1, \ldots, g_k\}$.*
3. *Afterwards we interpret $\langle G_i \cup f_{i+1} \rangle$ as the new input ideal of the algorithm, i.e. $\mathfrak{s}(g_j) = e_j$ for all $j = 1, \ldots, k$ and $\mathfrak{s}(f_{i+1}) = e_{k+1}$.*
4. *We also reconstruct all possible trivial syzygies as a starting point for the syzygy criterion.*

In general, this attempt gives a speedup up to $20-30\%$ and decreases the overall memory usage of the signature-based algorithm.

Since the above explained steps do not depend on the underlying ring or field structure, we can apply this technique to the Euclidean ring case, too. Luckily, the benefits are even bigger there: The number of polynomials decreases enormously, but also coefficient swell is often cut down. We illustrate this by the following small example:

EXAMPLE 3. *Let $I = \langle f_1, f_2, f_3 \rangle \subset \mathbb{Z}[x]$ with $f_1 = 4x^4$, $f_2 = 4x^3 + 2x^2 + 4x$, and $f_3 = 14x^5 + 3x$. Computing a Gröbner basis for I using a signature-based algorithm we first get an intermediate basis $G_2 = \{g_1, \ldots, g_{11}\}$ for $\langle f_1, f_2 \rangle$:*

		Polynomials	*Signatures*
g_1	$=$	$f_1 = 4x^4$	e_1
g_2	$=$	$f_2 = 4x^3 + 2x^2 + 4x$	e_2
g_3	$=$	$2x^3 + 4x^2$	xe_2
g_4	$=$	$6x^2 - 4x$	$2xe_2$
g_5	$=$	$-4x^2 - 8x$	$2x^2e_2$
g_6	$=$	$-16x^2$	$2x^2e_2$
g_7	$=$	$-10x^2 - 4x$	$2x^2e_2$
g_8	$=$	$2x^2 - 12x$	$2x^2e_2$
g_9	$=$	$-14x^2 - 12x$	$4x^2e_2$
g_{10}	$=$	$-2x^2 - 20x$	$4x^2e_2$
g_{11}	$=$	$-32x$	$6x^2e_2$

Starting with the next element f_3 with signature e_3 we have to deal with all possible pairs. Instead, we apply Technique 8 and reduce G_2 to:

		Polynomials	*(New) Signatures*
g_1	$=$	$2x^2 - 12x$	e_1
g_2	$=$	$-32x$	e_2

Here we have adjusted the corresponding signatures such that we can assume $\langle g_1, g_2, f_3 \rangle$ as input for the algorithm at module position 3. As the last step we have to remove the old syzygies (their signatures are no longer valid) and to reconstruct new known syzygies, namely the trivial ones $g_1e_2 - g_2e_1$, $g_1e_3 - g_3e_1$, and $g_2e_3 - g_3e_2$.

REMARK 9. *Syzygies found during the computations often tend to have bigger coefficients, thus even the removing of these old syzygies and replacing them with the new ones can have a benefit: We might have less syzygies to test when applying the syzygy criterion, but the new ones have the smallest possible coefficients w.r.t. $\prec$, a fact that should not be underestimated when computing over Euclidean rings.*

5. GCD-PAIR REPLACEMENTS

Another optimisation for computation over Euclidean rings is to exploit the GCD-pairs as much as possible since, in general, they help to keep the coefficient growth at a low. Due to the restriction of the signatures we have to adjust our GCD-pair handling a bit.

First we prove that we can replace an element to be reduced by a corresponding GCD-pair in a specific setting:

LEMMA 10. *In a signature-based algorithm over Euclidean rings we can exchange any element $\alpha \in \mathscr{G}$ by $\text{gpair}(\alpha, \beta)$ if $\beta \in \mathscr{G}$ such that*

$$\text{gpair}(\alpha,\beta) = (\pm 1)\alpha + ct\beta \text{ and } \mathfrak{s}(\text{gpair}(\alpha,\beta)) = (\pm 1)\mathfrak{s}(\alpha)$$

for some constant $c \in \mathscr{R}$ and some monomial $t \in \mathscr{P}$. Such a special GCD-pair is called a replacement GCD-pair (for α).

PROOF. Since the signature stays the same under module multiplication with a unit it is clear that there is no $\mathfrak{s}$-reduction suppressed that would be allowed when keeping $\alpha \in \mathscr{G}$. W.l.o.g. we assume that $\text{gpair}(\alpha, \beta) = \alpha - ct\beta$ and $\mathfrak{s}(\text{gpair}(\alpha,\beta)) = \mathfrak{s}(\alpha)$. Let δ be the S-pair between α and some $\gamma \in \mathscr{G}$ and let δ' denote the S-pair between $\text{gpair}(\alpha, \beta)$ and γ:

$$\begin{aligned} \delta &= c_\alpha t_\alpha \alpha - c_\gamma t_\gamma \gamma, \\ \delta' &= c'_\alpha t'_\alpha \text{gpair}(\alpha,\beta) - c'_\gamma t'_\gamma \gamma. \end{aligned}$$

for some constants $c_\alpha, c'_\alpha, c_\gamma, c'_\gamma \in \mathscr{R}$ and monomials $t_\alpha, t'_\alpha, t_\gamma, t'_\gamma \in \mathscr{P}$. By construction of $\operatorname{gpair}(\alpha,\beta) = \alpha + ct\beta$ we can identify $t_\alpha = t'_\alpha$ and $t_\gamma = t'_\gamma$. Moreover, by the definition of a GCD-pair we know that there exists $d \in \mathscr{R}$ such that $d = \frac{c_\alpha}{c'_\alpha} = \frac{c_\gamma}{c'_\gamma}$. But then it holds that

$$d\delta' = c_\alpha t_\alpha \alpha + dct\beta - c_\gamma t_\gamma \gamma = \delta + dct\beta.$$

So if we assume that $\overline{\delta'}$ reduces to zero, then clearly $d\overline{\delta'}$ does so, too. Since $\beta \in \mathscr{G}$ it then follows that also $\overline{\delta}$ reduces to zero.

Analogously one can proof the situation for δ and δ' being GCD-pairs and not S-pairs. □

REMARK 11. *Note that as mentioned in Remark 4 we only have the signatures, i.e. the lead terms of the module representations available during practical computation. Thus we have to ensure in Lemma 10 that* $\mathfrak{s}(\alpha) = \pm\mathfrak{s}(\operatorname{gpair}(\alpha,\beta))$ *since we cannot recover any cancellation of the signatures.*

Lemma 10 now enables us to optimize the $\mathfrak{s}$-reduction in a signature-based algorithm. There are three situations where we should look for a replacement of an element α by a fitting $\operatorname{gpair}(\alpha,\beta)$:

TECHNIQUE 12.
1. *Assume that the algorithm has chosen the next pair γ being some* $\operatorname{spair}(\alpha,\beta)$ *or* $\operatorname{gpair}(\alpha,\beta)$ *for some elements $\alpha, \beta \in \mathscr{G}$. Before starting the $\mathfrak{s}$-reduction of $\overline{\gamma}$ w.r.t. the intermediate basis we try to find a replacement GCD-pair for γ.*
2. *After the $\mathfrak{s}$-reduction of an element γ we try to replace it by a replacement GCD-pair. If so, we try to further reduce this replacement GCD-pair.*
3. *By Theorem 6 we cannot guarantee that the computation are done by increasing signature, thus it can happen that an element γ newly added to $\mathscr{G}$ generates a replacement GCD-pair for some other element α already in $\mathscr{G}$. In this situation we remove α and all pairs generated with α and generate new pairs with the replacement GCD-pair.*

We illustrate the usage of this technique by a small example.

EXAMPLE 4. *Let $I = \langle f_1, f_2\rangle \subset \mathbb{Z}[x,y]$ with $f_1 = 28y^2$ and $f_2 = 16x^2 + 7xy$. In the first steps of signature-based Gröbner basis algorithm, handling the next pair by minimal signature, the following elements are generated:*

		Polynomials	*Signatures*
g_1	=	$f_1 = 28y^2$	e_1
g_2	=	$f_2 = 16x^2 + 7xy$	e_2
g_3	=	$\operatorname{gpol}(g_1,g_2) = 4x^2y^2 + 14xy^3$	$2y^2e_2$
g_4	=	$\operatorname{spol}(g_1,g_2) = 21xy^3$	$7y^2e_2$
g_5	=	$\operatorname{gpol}(g_4,g_1) = -7xy^3$	$7y^2e_2$
g_6	=	$\operatorname{spol}(g_1,g_3) = 14xy^3$	$14y^2e_2$
g_7	=	$\operatorname{gpol}(g_4,g_3) = x^2y^3$	$7xy^2e_2$
g_8	=	$\operatorname{gpol}(g_2,g_6) = -2x^2y^3$	$14xy^2e_2$
g_9	=	$\operatorname{gpol}(g_4,g_2) = -x^2y^3$	$21xy^2e_2$

Without Technique 12 g_4 is added to the intermediate basis and corresponding pairs are generated (highlighted gray). These elements are all useless due to g_5 which represents the polynomial of the replacement GCD-pair for g_4. With Technique 12 enabled, g_4 is not added to the intermediate basis, but only g_5 and new pairs are generated with g_5.

Note that Example 4 also shows one drawback of signature-based computations over Euclidean rings: In a usual Buchberger algorithm only g_7 would be generated, due to the signature-conserving $\mathfrak{s}$-reductions we end up with g_7, g_8 and g_9 since we cannot recover a cancellation of the signatures during the reduction process.

6. HANDLING SIGNATURE DROPS

As we have seen in the last example of the previous section, signature-based computation with the restriction to $\mathfrak{s}$-reductions often introduce a computational overhead. On the other hand, we have seen in Section 3 that drops of the signature might occur even when applying only $\mathfrak{s}$-reductions. Assuming a pair $c_\alpha t_\alpha \alpha - c_\beta t_\beta \beta$ over a field the signature would be

$$\max\left\{c_\alpha t_\alpha \mathfrak{s}(\alpha), c_\beta t_\beta \mathfrak{s}(\beta)\right\}$$

where we can normalize these elements (so we can assume $c_\alpha = c_\beta = 1$) which leads to $t_\alpha \mathfrak{s}(\alpha) \neq t_\beta \mathfrak{s}(\beta)$ by Lemma 5. Over Euclidean rings, on the other hand, the corresponding signature is given via

$$\operatorname{lt}\left(c_\alpha t_\alpha \mathfrak{s}(\alpha) - c_\beta t_\beta \mathfrak{s}(\beta)\right).$$

Thus we might end up with a smaller signature that, see, for example, $\operatorname{spair}(\alpha, e_2)$ in Example 2 in Section 2. As we cannot control this behaviour due to Theorem 6 we can try to relax the concept of $\mathfrak{s}$-reductions a bit:

TECHNIQUE 13. *In the following we present a customized version of $\mathfrak{s}$-reduction will be called* optimistic $\mathfrak{s}$-reduction*. Assume that a signature-based Gröbner basis algorithm taking the next pair from the pair set by minimal possible signature starts to $\mathfrak{s}$-reduce the pair given by α*
1. *We $\mathfrak{s}$-reduce α w.r.t. the current basis.*
2. *If there exists a reducer β such that there exist $c \in \mathscr{R}$ and $t \in \mathscr{P}$ with $ct\operatorname{lt}\left(\overline{\beta}\right) = \operatorname{lt}(\overline{\alpha})$ and $\mathfrak{s}(\alpha - ct\beta) < \mathfrak{s}(\alpha)$ then we start a usual reduction without taking care of the signatures anymore. If no such reducer β exists then we go on and add $\overline{\alpha}$ to the intermediate Gröbner basis.*
3. *If this secondary reduction ends with 0, then α was useless and we can go on with the $\mathfrak{s}$-reduction of the next pair out of the pair set. If this reduction would not end with 0 then we have to stop the algorithm, since we cannot track signatures correctly from this step onwards.*

Technique 13 tries to mimic the situation of signature drops that we have to handle while building new pairs also in the context of $\mathfrak{s}$-reductions. It is optimistic in the sense that the corresponding reduction might end up with zero and we can keep the signature-based algorithm and the stability of the signatures alive. Note that the secondary reduction part only kicks in if there exists a reducer β such that the signature drops. If there exist reducers with higher signatures, then the signature-based algorithm adds α to the intermediate basis as usual. The core idea is to detect those situations in which the overhead when computing over Euclidean rings is introduced.

In our experimental results we have seen that keeping the signatures stable, i.e. not introducing a signature drop at all, is the most important factor for the efficiency of signature-based Gröbner basis algorithms over Euclidean rings.

Assume the signature-based Gröbner basis algorithm handles elements by increasing module position, so we assume that the algorithm currently handles elements of index i: When terminating the algorithm due to a signature drop we can use Technique 8 again:

We intrerreduce the intermediate basis with the element that introduced the signature drop,[4] apply new signatures to the elements and go on with the next step in the algorithm. Note that this time we have to regenerate all pairs that existed when we terminated the algorithm, i.e. all pairs with at least one generator that is coming from an element with signature index i.

Elements that had signature index $< i$ already generated a Gröbner basis of first $i-1$ generators, thus we clearly do not have to reconsider pairs where both generators had signature index $< i$.

In general, one would assume to sort these new input elements for the next incremental step by the monomial order given on $\mathscr{P}$. As it turns out, this might not be the best idea.

EXAMPLE 5. *Consider the ideal $I = \langle f_1, f_2 \rangle$ generated by $f_1 = 5y^3 + 8y^2 + 6y$ and $f_2 = 8x^2y + 9x^2$ in $\mathbb{Z}[x,y]$. In the following we give the first steps of a signature-based algorithm always taking the next pair from the pair set by minimal possible signature.*

Polynomials			*Signatures*
g_1	$=$	$f_1 = 5y^3 + 8y^2 + 6y$	e_1
g_2	$=$	$f_2 = 8x^2y + 9x^2$	e_2
g_3	$=$	$x^2y^3 - 6x^2y^2 - 18x^2y$	$2y^2e_2$
g_4	$=$	$-19x^2y^2 + 54x^2$	$5y^2e_2$
g_5	$=$	$2x^2y^2 - 51x^2y$	$10y^2e_2$
g_6	$=$	$-x^2y^2 - 81x^2y$	$15y^2e_2$
g_7	$=$	$171x^2y + 432x^2$	$40y^2e_2$
g_8	$=$	$-213x^2y$	$40y^2e_2$

The next element considered in the algorithm is $\mathrm{gpol}(g_4, g_5) = -x^2y^2 - 459x^2y + 54x^2$ *with signature* $95y^2e_2$.

The only possible reducer in the intermediate basis is f_6. Carrying out the reduction leads to the polynomial $-378x^2y + 54x^2$ which has decreased signature

$$95y^2e_2 - 15y^2e_2 = 80y^2e_2.$$

Thus we reached a signature drop and we can further reduce the above polynomial, taking signatures not into account, to a new element $g_9 = -165x^2y + 54x^2$. Since we cannot further reduce g_9 with the intermediate basis, we have to terminate the algorithm, add this polynomial and sort the intermediate basis.[5] We end up with the following new input $f_1, \dots, f_9$ for the signature-based algorithm:

Sorted Polynomials			*New Signatures*
$f_1 := g_1$	$=$	$5y^3 + 8y^2 + 6y$	e_1
$f_2 := g_2$	$=$	$8x^2y + 9x^2$	e_2
$f_3 := g_8$	$=$	$213x^2y$	e_3
$f_4 := g_9$	$=$	$165x^2y - 54x^2$	e_4
$f_5 := g_7$	$=$	$171x^2y + 432x^2$	e_5
$f_6 := g_6$	$=$	$x^2y^2 + 81x^2y$	e_6
$f_7 := g_5$	$=$	$2x^2y^2 - 51x^2y$	e_7
$f_8 := g_4$	$=$	$19x^2y^2 - 54x^2$	e_8
$f_9 := g_3$	$=$	$x^2y^3 - 6x^2y^2 - 18x^2y$	e_9

[4]If the signature drop happened during the generation of a new pair, we have to add the polynomial corresponding to this pair here, too.
[5]For the sake of simplicity we do not interreduce the intermediate basis in this example.

Clearly, g_1 is added as first element in order to keep the incremental nature of the computation itself stable. The other elements g_2 up to g_9 are sorted by degree reverse lexicographical order.

Remember that we have at the moment only a Gröbner basis for $\langle g_1 \rangle$ namely $\{g_1\}$. Thus we would start the next round of our signature-based Gröbner basis algorithm in order to compute a basis for $\langle g_1, g_2 \rangle$. We see that this would lead to exactly the same computation as we have done already. We would get an infinite loop, terminating and restarting always at the same step in the algorithm.

From Example 5 we see that using the monomial order for reordering the intermediate bases once terminating an incremental step of a signature-based Gröbner basis algorithm due to a signature drop is not a good idea. Instead, it turns out that often it is the best to move the element which introduced the signature drop to the front.

EXAMPLE 6 (EXAMPLE 5 REVISITED). *We have terminated in the very same situation as in Example 5. This time we put g_1 on the first position (from the previous incremental Gröbner basis for $\langle g_1 \rangle$). Next we put first g_9, the element which introduced the signature drop, afterwards the other remaining elements sorted by the monomial order:*

Sorted Polynomials			*New Signatures*
$f_1 := g_1$	$=$	$5y^3 + 8y^2 + 6y$	e_1
$f_2 := g_9$	$=$	$165x^2y - 54x^2$	e_2
$f_3 := g_2$	$=$	$8x^2y + 9x^2$	e_3
$f_4 := g_8$	$=$	$213x^2y$	e_4
$f_5 := g_7$	$=$	$171x^2y + 432x^2$	e_5
$f_6 := g_6$	$=$	$x^2y^2 + 81x^2y$	e_6
$f_7 := g_5$	$=$	$2x^2y^2 - 51x^2y$	e_7
$f_8 := g_4$	$=$	$19x^2y^2 - 54x^2$	e_8
$f_9 := g_3$	$=$	$x^2y^3 - 6x^2y^2 - 18x^2y$	e_9

Restarting now the signature-based Gröbner basis computation we first compute a basis for $\langle f_1, f_2 \rangle$:

Sorted Polynomials			*Signatures*
g_1	$=$	$f_1 = 5y^3 + 8y^2 + 6y$	e_1
g_2	$=$	$f_2 = 165x^2y - 54x^2$	e_2
g_3	$=$	$-318x^2y^2 - 198x^2y$	y^2e_2
g_4	$=$	$3x^2y^2 - 4,230x^2y$	$14y^2e_2$
g_5	$=$	$-16,614x^2y$	$55y^2e_2$
g_6	$=$	$3x^2y - 70,686x^2$	$715y^2e_2$
g_7	$=$	$-299,052x^2$	$3,025y^2e_2$
g_8	$=$	$-x^2y^3 + 620x^2y^2 + 17,954244x^2$	$2y^3e_2$
g_9	$=$	$x^2y^3 - 8,468x^2y^2 - 141,372x^2$	$28y^3e_2$

At this point we are done with the computation of the Gröbner basis for $\langle f_1, f_2 \rangle$, so we can apply Technique 8 and obtain an intermediate reduced basis. From this we go on computing the next incremental step that results in

		Polynomials	Signatures
h_1	=	$g_7 = 299,052x^2$	e_1
h_2	=	$g_1 = 5y^3 + 8y^2 + 6y$	e_2
h_3	=	$g_6 = 3x^2y - 70,686x^2$	e_3
h_4	=	$x^2y^3 - 620x^2y^2 - 11,124x^2$	e_4
h_5	=	$-x^2y + 212,067x^2$	e_5
h_6	=	$266,463x^2$	$3e_5$
h_7	=	$1,917x^2$	$165e_5$

This computation is done without a signature drop, next interreduction leads to

		Polynomials	New Signatures
q_1	=	$1,917x^2$	e_1
q_2	=	$5y^3 + 8y^2 + 6y$	e_2
q_3	=	$x^2y - 1,197x^2$	e_3

Going on with the next incremental steps, all further elements $\mathfrak{s}$-reduce to zero, so

$$G = \{q_1, q_2, q_3\}$$

is the final Gröbner basis for $I = \langle 5y^3 + 8y^2 + 6y, 8x^2y + 9x^2 \rangle$.

Putting element that introduced the signature drop up front ensures termination of our algorithm, but in general it breaks the polynomial monomial order for the next round of the signature-based Gröbner basis algorithm. For computation over Euclidean rings we have two choices:

1. Either we do not apply Technique 13. Then the computation might introduce a huge overhead of elements added to the intermediate basis since we do only allow $\mathfrak{s}$-reductions..
2. Or we apply Technique 13. Then we have to reorder the intermediate basis in a way not depending on the monomial order. Thus we might loose lower order elements that could be helpful for faster $\mathfrak{s}$-reductions in the upcoming iterative step of the algorithm.

7. HYBRID ALGORITHMS

In order to overcome the drawbacks of signature-based Gröbner basis computation over Euclidean rings, we combine the advantages (less zero reductions, getting new lead terms faster before signature drops) with a not signature-based Gröbner basis computation.

In the following let SBA denote any signature-based Gröbner basis algorithm that implements the techniques presented in this paper, and let BBA denote the usual not signature-based Buchberger algorithm. We present a hybrid approach combining both algorithms in Algorithm 1.

As input, HBA get besides the input ideal I and the monomial order $<$ also an integer n. n limits the **for** loop in line 2: n gives the number of calls to SBA if SBA has not finished with a Gröbner basis for I, but terminated due to a signature drop (see Section 6). In these situations SBA will return the intermediate (interreduced and sorted w.r.t. the signature drop) basis G and the flag "done" is set to **false**. If one of these SBA computations in line 3 terminates with the final Gröbner basis for I, "done" is set to **true** and HBA returns G. Otherwise, HBA goes on to line 8, and calls BBA with the already intermediate basis G, precomputed by SBA.

So we ensure that we restart SBA at most $n-1$ times before we go on with a not signature-based Gröbner basis computation.

Algorithm 1 Hybrid algorithm HBA for computing Gröbner bases over Euclidean rings

```
Input: Ideal $I = \langle f_1, \ldots, f_m \rangle \subset \mathscr{P}$, monomial order $<$, integer $n$
Output: Gröbner basis $G$ for $I$ w.r.t. $<$
1: $G \leftarrow \{f_1, \ldots, f_m\}$
2: for ($k = 1$; $k <= n$; $k++$) do
3:     (done, $G$) $\leftarrow$ SBA($G, <$)
4:     if (done) then
5:         return $G$
6:     end if
7: end for
8: $G \leftarrow$ BBA($G, <$)
9: return $G$
```

REMARK 14. *In all examples we tested we found that $n = 1$ is the best choice. Usually, if a signature drop happens, it makes no real sense to restart the signature-based computation since the stability (in either signature order or polynomial lead term order) is broken. For very few examples we found the nice behaviour of Example 5 where SBA can finish the overall Gröbner basis computation. In these examples SBA is much faster than BBA, but, at the moment, we do not have a good heuristic for classifying these examples.*

8. TIMINGS

In this section we present timings for our hybrid algorithm HBA compared to a usual Buchberger algorithm BBA. We implemented HBA in the computer algebra system SINGULAR, see [5]. It is available since version $4-1-0$. We investigated random examples over the integers. Note that due to Remark 14 HBA has nearly the same running time as SINGULAR's implementation of BBA called STD. This is due to the fact that very often the signature drop occurs quite early in the algorithm and we switch over to STD.

In Table 1 we present examples we found in which the HBA attempt is sometimes a lot faster than a usual BBA algorithm. The examples are available under [8]. All examples are chosen over the integers with three variables and the degree reverse lexicographical order. The third column "STD/HBA" states the factor of the running times. We also have examples with different number of variables that behave in the same way, but since the complexity of computation over the integers are increase hugely with the number of variables and terms, we restricted us here to the given set of parameters.

All computation are done on an Intel Xeon X5460 compute server with a 3.16GHz CPU and 64GB of RAM, the code itself is single-threaded and not parallelized at the moment.

These examples show that there exists a benefit for hybrid signature-based computations, what is still open is how to classify these examples. It seems that signature-based algorithms can be useful for finding good reducers fast in special situations. On the other hand, if they do not give any benefit like in Example 15 the overhead of the first signature-based computation is rather small. We see that the speedup achieved from a signature-based precomputation also varies by a huge factor.

9. FINITE RINGS & ZERO DIVISORS

In finite rings the computation of a strong Gröbner basis relies, besides S-polynomials and GCD-polynomials, also on handling so-called extended polynomials.

DEFINITION 6. *Let $\mathscr{R}$ be a principal ideal ring.*

1. *For an element $a \in \mathscr{R}$ the* annihilator *of a is defined by*

$$\mathrm{Ann}(a) = \langle 0 \rangle : \langle a \rangle.$$

Examples	STD	HBA	STD/ HBA
1	10.43	0.37	28.19
2	24.91	0.10	249.10
3	87.27	0.39	223.77
4	83.51	0.20	417.55
5	23,200.05	5,873.21	3.95
6	134.29	0.61	220.15
7	17,216.12	251.95	68.33
8	47.66	0.30	158.87
9	99.02	0.33	300.06
10	5,141.56	10.41	493.91
11	21,019.11	38.24	549.66
12	1,904.38	0.15	12,695.87
13	2,039.01	115.65	17.63
14	46.51	0.17	273.59
15	1,004.56	1,128.07	0.89
16	554.02	337.55	1.641

Table 1: STD vs. HBA (time in seconds)

2. *For* $f \in \mathscr{P}$ *the* extended polynomial *of* f *is defined by*

$$\operatorname{epol}(f) = d \cdot f,\ d \text{ generator of } \operatorname{Ann}(\operatorname{lc}(f)).$$

Note that if $\mathscr{R}$ has no zero divisors then $\operatorname{epol}(f) = 0$ for all $f \in \mathscr{P}$. Moreover, if $\operatorname{Ann}(\operatorname{lc}(f)) \neq \varnothing$ then it always holds that

$$\operatorname{epol}(f) = d \cdot f = d(f - \operatorname{lt}(f)),\ d \text{ generator of } \operatorname{Ann}(\operatorname{lc}(f)).$$

Extended polynomials are necessary to ensure correctness when computing strong Gröbner bases:

EXAMPLE 7. *Let* $I = \langle f \rangle \subset \mathbb{Z}_{10}[x,y]$ *where* $f = 5x^3 + 5xy + 3y$. *Not assuming extended polynomials the resulting Gröbner basis is just* $G = \{f\}$. *Sadly, the lead ideal generated by G is only* $L(G) = \langle 5x^3 \rangle$. *Looking at the lead ideal generated by I it is clear that, for example,* $\operatorname{lt}(2f) \in L(I)$ *where* $2f = 6y \in \mathbb{Z}_{10}[x,y]$. *Taking the extended polynomial into account we see that* $\operatorname{Ann}(\operatorname{lc}(f)) = \operatorname{Ann}(5) = \langle 2 \rangle$ *and thus we consider* $\operatorname{epol}(f) = 2f = 6y$ *in the algorithm and add it to G.*

The problem is that we have to consider the signatures of the polynomials in $(\mathbb{Z}_k[x])^m$ which means that we can now also encounter signature drops if the corresponding coefficients are zero divisors multiplied correspondingly. At the moment we do not have a good solution for this problem. In general, HBA boils down to a small signature-based precomputation (with a very early signature drop) together with a usual BBA afterwards. The intermediate basis from SBA usually does not provide any benefit for the following BBA computation. We often found that HBA is slower than BBA due to the overhead of calling SBA first.

10. CONCLUSION

We presented a first generalization for signature-based algorithms to compute Gröbner bases over Euclidean rings. We have shown the problem of signature drops and have given several attempts to handle this situation. Moreover, we have shown a range of examples for which a hybrid attempt of signature-based algorithms combined with the ususal Buchberger algorithm can be computed way faster then without signature-based precomputations. One important next task is to understand and classify these ideals, possibly applying a heuristic to the computation.

Clearly, an F4-style algorithm besides the usual Buchberger algorithm in SINGULAR, is one of the next steps in order to speed up computations over Euclidean rings in general. Moreover, deeper investigations on SBA and HBA concerning different module monomial orderings for the signatures and their effect on the signature drops have to be done. Losing the incremental behaviour of the algorithm might lead to new improvements for computations of Gröbner bases over Euclidean rings. Furthermore, computations over finite rings $\mathbb{Z}_m$ (see Section 9) might benefit from this step, too.

11. REFERENCES

[1] Buchberger, B. *Ein Algorithmus zum Auffinden der Basiselemente des Restklassenringes nach einem nulldimensionalen Polynomideal*. PhD thesis, University of Innsbruck, 1965.

[2] Buchberger, B. A Criterion for Detecting Unnecessary Reductions in the Construction of Gröbner Bases. In *EUROSAM '79, An International Symposium on Symbolic and Algebraic Manipulation*, volume 72 of *Lecture Notes in Computer Science*, pages 3–21. Springer, 1979.

[3] Buchberger, B. Gröbner Bases: An Algorithmic Method in Polynomial Ideal Theory. pages 184–232, 1985.

[4] Buchberger, B. An Algorithm for Finding the Basis Elements of the Residue Class Ring of Zero Dimensional Polynomial Ideal (English translation of Bruno Buchberger's PhD thesis. *Journal of Symbolic Computation*, 41(3-4):475 – 511, 2006.

[5] Decker, W., Greuel, G.-M., Pfister, G., and Schönemann, H. SINGULAR *4-1-0 — A computer algebra system for polynomial computations*, 2016. http://www.singular.uni-kl.de.

[6] C. Eder and J.-C. Faugère. A Survey on Signature-based Algorithms for Computing Gröbner Bases. *Journal of Symbolic Computation*, 80, Part 3:719 – 784, 2017.

[7] Eder, C. and Perry, J. F5C: A Variant of Faugère's F5 Algorithm with reduced Gröbner bases. *Journal of Symbolic Computation, MEGA 2009 special issue*, 45(12):1442–1458, 2010. dx.doi.org/10.1016/j.jsc.2010.06.019.

[8] Eder, E., Pfister, G., and Popescu, A. *Examples for HBA*, 2017. http://www.mathematik.uni-kl.de/~ederc/download/issac-2017.tar.gz.

[9] Faugère, J.-C. A new efficient algorithm for computing Gröbner bases without reduction to zero F5. In *ISSAC'02, Villeneuve d'Ascq, France*, pages 75–82, July 2002. Revised version from http://fgbrs.lip6.fr/jcf/Publications/index.html.

[10] Gao, S., Volny IV, F., and Wang, M. A new framework for computing Gröbner bases. *Mathematics of Computation, American Mathematical Society*, 85:449–465, 2016.

[11] Gebauer, R. and Möller, H. M. On an installation of Buchberger's algorithm. *Journal of Symbolic Computation*, 6(2-3):275–286, October/December 1988.

[12] Kandri-Rody, A. and Kapur, D. Computing a Gröbner basis of a polynomial ideal over a Euclidean domain. *Journal of Symbolic Computation*, 6(1):37 – 57, 1988.

[13] D. Lichtblau. Effective computation of strong Gröbner bases over Euclidean domains. *Illinois J. Math.*, 56(1):177–194, 2012.

[14] O. Wienand. *Algorithms for Symbolic Computation and their Applications - Standard Bases over Rings and Rank Tests in Statistics*. PhD thesis, 2011.