Computer Science Journal of Moldova, vol.**25**, no.3(75), 2017

Computing Comprehensive Gröbner Systems: A Comparison of Two Methods

Amir Hashemi, Benyamin M.-Alizadeh,
Mahdi Dehghani Darmian

Abstract

In this paper, we consider two main approaches to compute Gröbner bases for parametric polynomial ideals, namely the DisPGB algorithm developed by Montes [18] and the PGBMain proposed by Kapur, Sun and Wang [11]. The former algorithm creates new branches in the space of parameters during the construction of Gröbner basis of a given ideal in the polynomial ring of variables and the latter computes (at each iteration) a Gröbner basis of the ideal in the polynomial ring of the variables and parameters and creates new branches according to leading coefficients in terms of parameters. Therefore, the latter algorithm can benefit from the efficient implementation of Gröbner basis algorithm in each computer algebra system. In order to compare these two algorithms (in the same platform) we use the recent algorithm namely GVW due to Gao et al. [8] to compute Gröbner bases which makes the use of the F_5 criteria proposed by Faugère to remove superfluous reductions [6]. We show that there exists a class of examples so that an *incremental* structure on the DisPGB algorithm by using the GVW algorithm is faster than the PGBMain by applying the same algorithm to compute Gröbner bases. The mentioned algorithms have been implemented in `Maple` and experimented with a number of examples

Keywords: Comprehensive Gröbner systems, DisPGB algorithm, PGBMain algorithm, F_5 criteria, GVW algorithm.

1 Introduction

One of the most important tools in computer algebra is *Gröbner bases*. This concept along with the first algorithm to compute it, were introduced in 1965 by Buchberger in his PhD thesis (see [3]). His two criteria and the implementation methods [4] transformed Gröbner bases to a powerful tool to tackle many important problems in polynomial ideals theory. However, Buchberger's algorithm was not efficient in practice for large polynomial systems. In 1983, Lazard described a new algorithm to compute Gröbner bases, using linear algebra techniques [14]. In 1988, Gebauer and Möller have installed Buchberger's two criteria on Buchberger's algorithm in an efficient manner (see [9]). In 1999, Faugère described his F_4 algorithm to compute Gröbner bases (see [5]). This algorithm (which is an efficient algorithm based on [9], [14]) exploits fast linear algebra on sparse matrices, and has been implemented in `Maple` and `Magma`. In 2002, Faugère has described the *F_5 algorithm*; a new incremental algorithm which makes the use of the *F_5 criteria* to compute Gröbner bases [6] (see also [17]). Ars and Hashemi [1] proposed a non-incremental version of this algorithm by defining new orderings on the signatures to make it independent from the order of the input polynomials. Gao et al. [7] presented G^2V; a variant of the F_5 algorithm which is simpler and more efficient than F_5. Finally, Gao et al. [8] proposed a new framework more general than the G^2V algorithm, namely GVW to compute simultaneously Gröbner bases for an ideal and its syzygy module.

The concept of comprehensive Gröbner bases can be considered as an extension of Gröbner bases of polynomials over fields to polynomials with parametric coefficients. This extension plays an important role in the applications such as constructive algebraic geometry, robotics, electrical network, automatic theorem proving and so on (see e.g. [15], [16], [18], [19]). *Comprehensive Gröbner bases* and *comprehensive Gröbner systems* (for simplification, we employ the term CGS to refer to comprehensive Gröbner system) were introduced in 1992 by Weispfenning [24]. He proved that any parametric polynomial ideal has a finite CGS and described an algorithm to compute it. In 2002, Montes [18] proposed a more efficient algorithm namely DisPGB for computing CGSs. Suzuki and Sato [21] provided an important improvement for computing CGSs using only computations of reduced Gröbner bases in polynomial rings over ground fields (we subsequently refer to this algorithm as the Suzuki-Sato algorithm). In 2010, Kapur et al. [11] by combining Weispfenning's algorithm [24] with the Suzuki-Sato algorithm, gave a new algorithm (that we refer to as PGBMain algorithm) for computing CGSs (see also [12], [13]). Finally, Montes and Wibmer in [20] presented the GröbnerCover algorithm (see [23]) which computes a finite partition of the space of parameters into locally closed subsets together with polynomial data, from which the reduced Gröbner basis for a given parameter point can immediately be determined.

It is worth noting that PGBMain at each iteration computes the Gröbner basis over a polynomial ring in the variables and parameters. Therefore, it makes the use of a Gröbner basis function in each computer algebra system. On the other hand, DisPGB reduces the computation in a polynomial ring of only variables by creating new branches when a new polynomial with an undecidable coefficient is constructed. So a natural question arises: *Which of these two algorithms is more efficient in practice?* In this paper, we consider this question by proposing an incremental structure on DisPGB by applying GVW equipped with the F_5 criteria. We have implemented in `Maple` this algorithm and also PGBMain by using GVW as the engine of Gröbner bases computation. We compare the performance of these algorithms on a number of polynomial ideals by showing that there exists a class of ideals for which our new variant of DisPGB is more efficient than PGBMain. We shall mention that due to the structure of PGBMain, its outputs in general have less number of branches than DisPGB.

Now, we give the structure of the paper. Section 2 contains the basic definitions and notations related to CGSs, and a short description of DisPGB. In Section 3, we present briefly GVW. Section 4 is devoted to the description of our new algorithm namely GVWDisPGB for computing CGSs. In Section 5, we show the performance of this algorithm w.r.t. our implementation of PGBMain in `Maple` and the function `cgsdr` of `Singular` via some examples.

2 Comprehensive Gröbner systems and DisPGB algorithm

In this section, we recall the basic definitions and notations concerning CGSs, and describe briefly the DisPGB algorithm.
Let $R = K[\mathbf{x}]$ be a polynomial ring, where $\mathbf{x} = x_1, \ldots, x_n$ is the sequence of variables and K an arbitrary field. Let $I = \langle f_1, \ldots, f_k \rangle$ be the ideal of R generated by the polynomials $f_1, \ldots, f_k$. Also, let $f \in R$ and let $\prec$ be a monomial ordering on R. The *leading monomial* of f is the greatest monomial (w.r.t. $\prec$) appearing in f, and we denote it by $\mathrm{LM}(f)$. The *leading coefficient* of f, denoted by $\mathrm{LC}(f)$, is the coefficient of $\mathrm{LM}(f)$. The *leading term* of f is $\mathrm{LT}(f) = \mathrm{LC}(f)\mathrm{LM}(f)$. The *leading term ideal* of I is defined to be

$$\mathrm{LT}(I) = \langle \mathrm{LT}(f) \ \mid \ f \in I \rangle.$$

A finite set $G = \{g_1, \ldots, g_k\} \subset I$ is called a *Gröbner basis* of I w.r.t. $\prec$ if $\mathrm{LT}(I) = \langle \mathrm{LT}(g_1), \ldots, \mathrm{LT}(g_k) \rangle$. For more details, we refer to [2], pages 213–214.

Now consider $F = \{f_1, \ldots, f_k\} \subset S = K[\mathbf{a}, \mathbf{x}]$, where $\mathbf{a} = a_1, \ldots, a_m$ is the sequence of parameters. Let $\prec_{\mathbf{x}}$ (resp. $\prec_{\mathbf{a}}$) be a monomial ordering involving the x_i's (resp. a_i's). We also need a compatible elimination product ordering $\prec_{\mathbf{x},\mathbf{a}}$. It is defined as follows: For all $\alpha, \gamma \in \mathbb{Z}^n_{\geq 0}$ and $\beta, \delta \in \mathbb{Z}^m_{\geq 0}$

$$\mathbf{x}^\gamma \mathbf{a}^\delta \prec_{\mathbf{x},\mathbf{a}} \mathbf{x}^\alpha \mathbf{a}^\beta \ \text{ iff } \ \begin{cases} \mathbf{x}^\gamma \prec_{\mathbf{x}} \mathbf{x}^\alpha & \text{or} \\ \mathbf{x}^\gamma = \mathbf{x}^\alpha & \text{and} \quad \mathbf{a}^\delta \prec_{\mathbf{a}} \mathbf{a}^\beta. \end{cases}$$

Now, we recall the definition of a CGS for a parametric ideal.

Definition 1. *Let $G = \{(G_i, N_i, W_i)\}_{i=1}^{\ell}$ be a finite set of triples, where $N_i, W_i \subset K[\mathbf{a}]$ and $G_i \subset S$ are finite for $i = 1, \ldots, \ell$. The set G is called a CGS for $\langle F \rangle$ w.r.t. $\prec_{\mathbf{x},\mathbf{a}}$ if for any specialization $\sigma : K[\mathbf{a}] \to \bar{K}$ with $\bar{K}$ the algebraic closure of K there exists i such that the following conditions hold*

- *$\sigma(G_i) \subset \bar{K}[\mathbf{x}]$ is a Gröbner basis for $\sigma(\langle F\rangle) \subset \bar{K}[\mathbf{x}]$ w.r.t. $\prec_{\mathbf{x}}$*
- *$\sigma(p) = 0$ for each $p \in N_i$ and $\sigma(q) \neq 0$ for each $q \in W_i$.*

For each i, the set N_i (resp. W_i) is called a (resp. non-) null conditions set. Each pair (N_i, W_i) is called a *specification* (for a homomorphism σ if both the conditions in the above definition are satisfied).

Now, we describe shortly Montes DisPGB algorithm to compute CGSs for parametric ideals (see [15], [18]). The main idea of DisPGB is based on discussing the nullity or not w.r.t. a given specification (N, W) of the leading coefficients of the polynomials appearing at each step (this process is performed by NewCond subalgorithm). Let us consider a set $F \subset S$ of parametric polynomials. For a given polynomial $f \in F$, and a given specification (N, W), NewCond is called. Three cases are possible: If LC(f) specializes to zero w.r.t. (N, W), we replace f by $f - \text{LT}(f)$, and then start again. If LC(f) specializes to a non-zero element, we continue with the next polynomial in F. Otherwise (if LC(f) is not decidable), the subalgorithm Branch is called to create two supplementary cases by assuming LC$(f) = 0$ and LC$(f) \neq 0$. Therefore, two new disjoint branches with the specifications $(N \cup \{\text{LC}(f)\}, W)$ and $(N, W \cup \{\text{LC}(f)\})$ will be made. This procedure will continue until every polynomial in F has a non-null leading coefficient w.r.t. the current specification. Then, we proceed with CondPGB: This algorithm receives as input a set of parametric polynomials and a specification (N, W) and using Buchberger's algorithm, it creates new polynomials. When a new polynomial is generated, NewCond verifies whether its leading coefficient gives a new condition or not. If a new condition is found it stops, and Branch is called to make two new disjoint branches. Otherwise, this continues and computes a Gröbner basis for $\langle F\rangle$, according to the current specification. The collection of these bases, gives a CGS for $\langle F\rangle$.

3 F$_5$ criteria and GVW algorithm

This section aims to present the F$_5$ theory. After recalling some notations and definitions (used also in the next section), we state the main theorem of [6] which forms the basis of the F_5 algorithm. Finally, we present briefly the GVW algorithm following [8].

Let $R = K[\mathbf{x}]$ be a polynomial ring, where $\mathbf{x} = x_1, \ldots, x_n$ is the sequence of variables, K is an arbitrary field and $I = \langle f_1, \ldots, f_k \rangle$ is the ideal of R generated by the polynomials $f_1, \ldots, f_k$. Let R^k be a k-dimensional R-module and $\mathbf{f_1}, \ldots, \mathbf{f_k}$ its canonical basis. A module monomial is an element of R^k of the form $m\mathbf{f_i}$, where $m \in R$ is a monomial. Given two module monomials $m\mathbf{f_i}$ and $m'\mathbf{f_j}$, one can extend a monomial ordering $\prec$ on R to a module monomial ordering on R^k in different ways. In [8] the authors proposed four different module monomial orderings. Below, we recall one of them under which GVW closely corresponds to the G^2V algorithm presented in [7].

$$m\mathbf{f}_i < m'\mathbf{f}_j \ \text{ if } \ \begin{cases} j < i & \text{or} \\ i = j & \text{and} \quad m \prec m'. \end{cases}$$

For an element $\mathbf{g} = \sum_{i=1}^{k} g_i \mathbf{f_i} \in R^k$, we define the index of $\mathbf{g}$, index($\mathbf{g}$) to be the lowest integer i such that $g_i \neq 0$. Let index($\mathbf{g}$) $= i_0$, then we call $\mathrm{LM}(g_{i_0})\mathbf{f_{i_0}}$ the *module leading monomial* of $\mathbf{g}$ and denote it by MLM($\mathbf{g}$). Also, we use LM($\mathbf{g}$) to denote $\mathrm{LM}(\sum_{i=1}^{k} g_i f_i)$.

The elements of the form $r = (m\mathbf{f}_i, f) \in A = R^k \times R$, where m is a monomial, i an integer and f a polynomial are called labelled polynomials. $\mathrm{S}(r) = m\mathbf{f}_i$ is called the signature part of r and $\mathrm{poly}(r) = f$ the polynomial part of r. Denote ψ the map $\psi : R^k \to R$ so that $\psi(g_1, \ldots, g_k) = g_1 f_1 + \cdots + g_k f_k$. A labelled polynomial $r = (\mathrm{S}(r), \mathrm{poly}(r))$ is called admissible if there exists $g \in R^k$ such that $\psi(g) = \mathrm{poly}(r)$ and $\mathrm{MLM}(g) = \mathrm{S}(r)$. We define the following operations on labelled polynomials: Let $r = (m\mathbf{f}_i, f)$ be a labelled polynomial, u a monomial and c a constant. Then we define $ur = (um\mathbf{f}_i, uf)$ and $cr = (m\mathbf{f}_i, cf)$. These definitions obviously preserve the admissibility. The special reduction of F_5 also preserves it and ensures that during a Gröbner basis computation by F_5, the labelled polynomials always take the minimal possible signature and remain admissible. We also need the following definitions to state the main theorem.

Definition 2 (F_5 criterion)**.** *An admissible labelled polynomial* $r =$

$(m\mathbf{f_i}, f)$ is called normalized *if we have $m \notin \mathrm{LT}(\langle f_{i+1}, \ldots, f_k\rangle)$. A pair (r, s) of admissible labelled polynomials is normalized if ur and vs are normalized, where $r = (m\mathbf{f_i}, f)$, $s = (m'\mathbf{f_j}, g)$, $u = \frac{\mathrm{lcm}(\mathrm{LM}(f),\mathrm{LM}(g))}{\mathrm{LM}(f)}$ and $v = \frac{\mathrm{lcm}(\mathrm{LM}(f),\mathrm{LM}(g))}{\mathrm{LM}(g)}$.*

Faugère has described F_5 as an incremental algorithm to use the F_5 criterion, i.e. to compute the Gröbner basis of I, it computes respectively the Gröbner bases of the ideals

$$\langle f_k\rangle, \langle f_{k-1}, f_k\rangle, \ldots, \langle f_1, \ldots, f_k\rangle.$$

In the following, we define the concept of t-representation for labelled polynomials, imposing additional conditions on the signatures (see [2, page 219]).

Definition 3. *Let $P \subset A$ be a finite set of labelled polynomials, and $r, t \in A$ two labelled polynomials with $\mathrm{poly}(r) = f$, where $f \neq 0$. We say that $f = \sum_{p_i \in P} h_i \mathrm{poly}(p_i)$ is a t-*representation *of r w.r.t. P if for all $p_i \in P$ with $\mathrm{poly}(p_i) \neq 0$ we have*

$$\mathrm{LM}(h_i)\mathrm{LM}(\mathrm{poly}(p_i)) \preceq \mathrm{LM}(\mathrm{poly}(t)) \quad \text{and} \quad \mathrm{LM}(h_i)\mathrm{S}(p_i) < \mathrm{S}(r).$$

This property is denoted by $r = \mathcal{O}_P(t)$. We write $s = o_P(t)$ if there exists labelled polynomial $t' \in A$ satisfying $\mathrm{S}(t') < \mathrm{S}(t)$ and $\mathrm{LM}(\mathrm{poly}(t')) \prec \mathrm{LM}(\mathrm{poly}(t))$ such that $s = \mathcal{O}_P(t')$.

Let $f, g \in R$ be two polynomials. The *S-polynomial* of f and g is defined as:

$$\mathrm{Spoly}(f, g) = \frac{\mathrm{lcm}(\mathrm{LM}(f), \mathrm{LM}(g))}{\mathrm{LT}(f)} f - \frac{\mathrm{lcm}(\mathrm{LM}(f), \mathrm{LM}(g))}{\mathrm{LT}(g)} g.$$

Let $r = (\mathrm{S}(r), f)$ and $s = (\mathrm{S}(s), g)$ be two admissible labelled polynomials such that $v\mathrm{S}(s) < u\mathrm{S}(r)$ with $u = \frac{\mathrm{lcm}(\mathrm{LM}(f),\mathrm{LM}(g))}{\mathrm{LM}(f)}$ and $v = \frac{\mathrm{lcm}(\mathrm{LM}(f),\mathrm{LM}(g))}{\mathrm{LM}(g)}$. Then, we define $\mathrm{Spoly}(r, s) = (u\mathrm{S}(r), \mathrm{Spoly}(f, g))$.

Theorem 1. ([6]) *Let $I = \langle f_1, \ldots, f_k\rangle \subset R$. Let $G \subset A$ be a finite set of admissible labelled polynomials such that*

- *for every i, we have $f_i = \mathrm{poly}(r_i)$ for some $r_i \in G$,*
- *for each $(r_i, r_j) \in G \times G$ which is normalized, $\mathrm{Spoly}(r_i, r_j)$ is either zero or equal to $o_G(u_s r_s)$, where $u_s = \frac{\mathrm{lcm}(\mathrm{LM}(\mathrm{poly}(r_i)),\mathrm{LM}(\mathrm{poly}(r_j)))}{\mathrm{LM}(\mathrm{poly}(r_s))}$ for $s \in \{i, j\}$.*

Then the set $\{\mathrm{poly}(r) \mid r \in G\}$ is a Gröbner basis for I.

Faugère in the F_5 algorithm has used another criterion, namely IsRewritten criterion, to detect more useless critical pairs, however he has not declared it explicitly in [6]. We recall this criterion and refer to [10] for more details.

Definition 4 (IsRewritten criterion)**.** *With the above notations, let $u \in R$ be a monomial and $r = (m\mathbf{f_i}, f)$ an admissible labelled polynomial. Then, the pair $[u, r]$ is called* rewritable *if there exists an admissible labelled polynomial $r' = (m'\mathbf{f_i}, f')$ computed after r, i.e. $\mathrm{S}(r) < \mathrm{S}(r')$, such that m' divides um. A pair (r, s) of admissible labelled polynomials is rewritable if $[u, r]$ or $[v, s]$ is rewritable, where $r = (m\mathbf{f_i}, f)$, $s = (m'\mathbf{f_j}, g)$, $u = \frac{\mathrm{lcm}(\mathrm{LM}(f),\mathrm{LM}(g))}{\mathrm{LM}(f)}$ and $v = \frac{\mathrm{lcm}(\mathrm{LM}(f),\mathrm{LM}(g))}{\mathrm{LM}(g)}$.*

As the following proposition yields, if a critical pair is rewritable, its S-polynomial has a standard representation w.r.t. the last computed Gröbner basis, and therefore the F_5 algorithm deletes all such pairs.

Proposition 1. ([10]) *Let $I = \langle f_1, \ldots, f_k \rangle \subset R$ be an ideal. Let r_i and r_j be two labelled polynomials treated during an execution of the F_5 algorithm for computing the Gröbner basis of I. If (r_i, r_j) is rewritable, then $\mathrm{Spoly}(r_i, r_j)$ is either zero or equal to $o_G(u_s r_s)$, where $u_s = \frac{\mathrm{lcm}(\mathrm{LM}(\mathrm{poly}(r_i)),\mathrm{LM}(\mathrm{poly}(r_j)))}{\mathrm{LM}(\mathrm{poly}(r_s))}$ for $s \in \{i, j\}$ (and so the pair (r_i, r_j) can be omitted).*

Throughout this paper, by the F_5 criteria we mean F_5 criterion and IsRewritten criterion. The main problem with the F_5 algorithm is that it is difficult to both understand and implement. Gao et al. in [8] presented the GVW algorithm which seems to be simpler and more efficient than the F_5 algorithm. That is why, we use this algorithm to apply the F_5 criteria on DisPGB.

To explain more precisely the structure of GVW, let us suppose that we are going to compute a Gröbner basis for the ideal $\langle f_1, \dots, f_k \rangle$ with respect to a monomial ordering $\prec$. The main difference of GVW and F_5 relies on the non-incremental structure of GVW which makes the use of different kinds of module monomial orderings. Let us consider e.g. module monomial ordering $<$ defined as above (remark that GVW endowed with this ordering is equivalent to G^2V [7]). Without lose of generality, suppose that for all labelled polynomials $(\mathbf{u}, v)$, $\mathrm{LC}(v) = 1$. Given two labelled polynomials $p_1 = (\mathbf{u}_1, v_1)$ and $p_2 = (\mathbf{u}_2, v_2)$, the J-pair of p_1 and p_2 is the new pair $t_i p_i$, where $t_i = \frac{\mathrm{lcm}(\mathrm{LM}(v_1),\mathrm{LM}(v_2))}{\mathrm{LM}(v_i)}$ and $t_i v_i = \max_{<}\{t_1\mathbf{u}_1, t_2\mathbf{u}_2\}$, provided that $t_1\mathbf{u}_1 \neq t_2\mathbf{u_2}$.

At the first step, GVW begins with the initial set of J-pairs $\{(\mathbf{f}_1, f_1), \dots, (\mathbf{f}_k, f_k)\}$. It takes in each step the smallest J-pair (w.r.t. signature) and repeatedly performs only *regular top reductions* until it is no longer regular top reducible. A labelled polynomial $(\mathbf{u}_1, v_1)$ is top reducible by $(\mathbf{u}_2, v_2)$ if there exists a monomial $t \in R$ such that $\mathrm{LM}(v_1) = t\mathrm{LM}(v_2)$ and $t\mathbf{u}_2 < \mathbf{u}_1$. The corresponding top reduction is

$$(\mathbf{u}_1, v_1) - t(\mathbf{u}_2, v_2) = (\mathbf{u}_1, v_1 - tv_2).$$

If $t\mathbf{u}_2 = \mathbf{u}_1$, the top reduction is called *super*, otherwise it is called *regular*. Let $(\mathbf{u}, v)$ be the result of the reduction of a labelled polynomial. If $v \neq 0$, we add $(\mathbf{u}, v)$ to the current Gröbner basis, and form the new J-pairs. Otherwise, GVW uses $\mathbf{u}$ to delete useless J-pairs: For any labelled polynomial $(\mathbf{u}', v')$, if $t\mathbf{u} = \mathbf{u}'$ for some monomial t, then we can discard $(\mathbf{u}', v')$, provided that $t\mathrm{LM}(v) \prec \mathrm{LM}(v')$. Indeed, this is a special case of super top reduction, where $(\mathbf{u}', v')$ is super top reducible by $(\mathbf{u}, 0)$. Furthermore, a J-pair $(\mathbf{u}, v)$ is called *covered* by G if there is a pair $(\mathbf{u}', v') \in G$ so that $\mathbf{u}'$ divides $\mathbf{u}$ and $t\mathrm{LM}(v') \prec \mathrm{LM}(v)$ (strictly smaller), where $t = \mathbf{u}'/\mathbf{u}$ is a monomial.

Remark 1. *The relation of the criteria used in GVW with F_5 criterion (Theorem 1) and IsRewritten criterion (Proposition 1) is illustrated in [8, Corollaries 2.5 and 2.6], respectively. Furthermore the correctness and termination of GVW are proved in [8, Theorem 3.1].*

4 Description of the new algorithm

In this section, we show how to combine the GVW algorithm with the DisPGB algorithm to compute CGSs for parametric ideals. For this, we use the improved version of DisPGB described in [15], and an incremental structure on DisPGB to be able to apply the F_5 criteria. More precisely, let $I = \langle f_1, \dots, f_k \rangle \subset K[\mathbf{a}, \mathbf{x}]$ be a parametric ideal, where $\mathbf{x} = x_1, \dots, x_n$ is the sequence of variables and $\mathbf{a} = a_1, \dots, a_m$ is the sequence of parameters. Let $\prec_{\mathbf{x}}$ (resp. $\prec_{\mathbf{a}}$) be a monomial ordering involving the x_i's (resp. a_i's). Then, to compute a CGS for I, we compute CGSs of the ideals $\langle f_k \rangle, \langle f_{k-1}, f_k \rangle, \dots, \langle f_1, \dots, f_k \rangle$ respectively and for each i, we use the CGS of $\langle f_{i+1}, \dots, f_k \rangle$ to compute a CGS for $\langle f_i, \dots, f_k \rangle$.

Example 1. *In this simple example, we show how an incremental structure may be used to compute a CGS for an ideal. Let $I = \langle ax + 1, by + 1 \rangle \subset K[\mathbf{a}, \mathbf{x}]$, where $\mathbf{a} = a, b$ and $\mathbf{x} = x, y$. We compute first a CGS for the ideal $\langle by + 1 \rangle$ which is equal to $\{(\{1\}, \{b\}, \{\}), (\{by + 1\}, \{\}, \{b\})\}$. Now, we will discuss the addition of $ax + 1$ to each member of this system according to nullity or not of a. It follows the following CGS for I:*

$$\{(\{1\}, \{b\}, \{a\}), (\{1\}, \{a, b\}, \{\}), (\{1\}, \{a\}, \{b\}), (\{by + 1, ax + 1\}, \{\}, \{b, a\})\}.$$

We describe now the main algorithm GVWDisPGB which computes incrementally a CGS for a given ideal.

Algorithm 1 GVWDisPGB

```
Require: $F$ : finite subset of $S$
Ensure: A CGS for $\langle F \rangle$
  global: List, Grob, JP
  List:=Null
  $\{f_1, \dots, f_k\}$:=InterReduce($F, \prec_{\mathbf{x},\mathbf{a}}$)
  Branch($(1, f_k), \{ \}, \{ \}, \{ \}, \{ \}, \{ \}$)
  for i from $k - 1$ to 1 do
    IncDisPGB($f_i$,{List})
  end for
  Return (List)
```

Note that the function `InterReduce`$(F, \prec_{\mathbf{x},\mathbf{a}})$ inter-reduces a list of polynomials F w.r.t. $\prec_{\mathbf{x},\mathbf{a}}$; i.e. every polynomial in F must be divided by the remaining elements of F such that at the output no monomial of any polynomial of F is divisible by the leading monomials of the other polynomials in F. In the above algorithm, List (resp. JP) is a global variable in which (and at each iteration) we save the computed CGS (resp. set of J-pairs). That is why, at the beginning of each iteration we must keep them null (see the next subalgorithm). Indeed, Branch calculates a CGS for $\langle f_k \rangle$ and save it in List. Then, for any i between $k-1$ and 1, IncDisPGB computes a CGS for the ideal $\langle f_i, \dots, f_k \rangle$ (using the CGS of $\langle f_{i+1}, \dots, f_k \rangle$ which has already been computed) and saves it in List. Thus, at the end, List is a CGS for the ideal $\langle f_1, \dots, f_k \rangle$. Now, we describe IncDisPGB.

Algorithm 2 IncDisPGB

```
Require: $\begin{cases} f_i : & \text{a polynomial with } 1 \le i \le k-1 \\ \{G_1, \dots, G_t\} : & \text{a CGS for } \langle f_{i+1}, \dots, f_k \rangle \end{cases}$
Ensure: A CGS for $\langle f_i, \dots, f_k \rangle$
  L:=Null
  for $j$ from 1 to $t$ do
    JP:={ }, List:=Null
    $(\text{Grob}, N, W) := G_j$
    $f := \overline{f_i}^N$ (the remainder of the division of $f_i$ by $N$)
    if $f = 0$ then
      List:=List,$G_j$
    else
      Branch$((1, f), \text{Grob}, N, W, \{\ \}, \{(0, g) \mid g \in \text{Grob}\})$
    end if
    L:=L,List
  end for
  List:=L
```

Algorithm 3 BRANCH

```
Require: $\begin{cases} (u,f): & \text{labelled polynomial} \\ B: & \text{specializing basis} \\ N,W: & \text{where } (N,W) \text{ is an specification} \\ JP: & \text{set of J-pairs} \\ R: & \text{set of computed labelled polynomials } (v,g) \text{ s.t.} g \in B \end{cases}$
Ensure: It stores the refined $(B', N', W', JP', R')$, and either creates two
  new vertices when necessary or marks the vertex as terminal
  $(N,W)$:=CANSPEC(N,W)
  $(cd, f, N, W)$ :=NEWCOND$(f, N, W)$
  if $cd = \{\ \}$ then
    $(test, (u', f'), B', N', W', JP', R')$ :=CONDPGB$((u,f), B, N, W, JP, R)$
    if test then
      if $JP = \{\ \}$ then
        List:=List,$(B', N, W)$
      end if
    else
      BRANCH$((u', f'), B', N', W', JP', R')$
    end if
  else
    BRANCH$((u,f), B, N \cup cd, W, JP, R)$
    BRANCH$((u,f), B, N, W \cup cd, JP, R)$
  end if
```

To clarify BRANCH, suppose that it receives a polynomial f and a specification (N, W). If $\mathrm{LC}(f)$ is not decidable w.r.t. (N, W), this subalgorithm creates two supplementary cases by adding $\mathrm{LC}(f) = 0$ and $\mathrm{LC}(f) \neq 0$ to the set of null and non-null conditions set respectively. Therefore, two new disjoint branches with the specifications $(N \cup \{\mathrm{LC}(f)\}, W)$ and $(N, W \cup \{\mathrm{LC}(f)\})$ will be made. We explain here the significance of the variables B and cd in BRANCH. The variable B contains all polynomials related to the corresponding branch which will be completed, and at the end of the branch it will be the Gröbner basis for the corresponding specification. We explain now the variable cd. When we call NEWCOND(f, N, W), if $\mathrm{LC}(f)$ is decidable w.r.t the specification (N, W), it returns $(\emptyset, f - \mathrm{LT}(f), N, W)$ in the case that $\mathrm{LC}(f)$ specializes to zero w.r.t. (N, W), and it returns

$(\emptyset, f, N, W)$ in the case that $\mathrm{LC}(f)$ does not specialize to zero w.r.t. (N, W). Otherwise, it returns (cd, f', N, W), where cd contains one of the non-decidable factors (w.r.t (N, W)) of $\mathrm{LC}(f)$. We describe below the NewCond subalgorithm in which FacVar factors a polynomial in parameters. Remark that $\textsc{FacVar}(\mathrm{LC}(f')) \setminus W'$ returns only one factor of $\mathrm{LC}(f')$.

Algorithm 4 NewCond

```
Require: { f :      a parametric polynomial
         { N, W :   where (N, W) is an specification
Ensure:  { cd :       set of a new condition
         { f' :       a parametric polynomial
         { N', W' :   where (N', W') is an specification
  $f' := f$
  test:=true
  $N' := N$
  while test do
    if $\mathrm{LC}(f') \in \sqrt{\langle N' \rangle}$ then
      $N' :=$ a Gröbner basis for $\langle N', \mathrm{LC}(f') \rangle$ w.r.t. $\prec_{\mathbf{a}}$
      $f' := f' - \mathrm{LM}(f)$
    else
      test:=false
    end if
    $f' := \overline{f'}^{N'}$
    $W' := \{\overline{w}^{N'} \mid w \in W\}$
    $cd := \textsc{FacVar}(\mathrm{LC}(f')) \setminus W'$
  end while
  Return $(cd, f', N', W')$
```

We shall note that $\sqrt{I}$ denotes the radical of I, i.e., the set of all elements for which some positive power lies in I. By Hilbert Nullstellensatz, if a polynomial over an algebraically closed field vanishes on the vanishing set of an ideal, then a power of the polynomial belongs to the ideal, see [2, Theorem 7.40]. For a radical membership test (which has been used in the algorithm), we refer to [2, page 268]. The next algorithm is a variant of the GVW algorithm (to apply the F_5 criteria) for parametric ideals.

Algorithm 5 CondPGB

```
Require: $\begin{cases} (u,f): & \text{labelled polynomial} \\ B: & \text{specializing basis} \\ N,W: & \text{where } (N,W) \text{ is an specification} \\ JP: & \text{set of J-pairs} \\ R: & \text{set of labelled polynomials} \end{cases}$
Ensure: (flag,$(v,g)$, $B', N', W', JP', R'$) with flag=true when $f$ does not generate no new condition and no new JPair and $B' = B \cup \{f\}$ is a Gröbner basis for the corresponding branch; otherwise, flag=false and $B, N, W, JP, R$ are updated
  $B := \overline{B}^N$; (if a polynomial in $B$ is reduced to zero, then remove all J-pairs in $JP$ containing this polynomial)
  $R := \{(v, \overline{g}^N) \mid (v,g) \in R\}$
  $LM := \{\mathrm{LM}(g) \mid g \in \mathrm{Grob}\}$
  $f := \overline{f}^N$
  $f := \overline{f}^{\mathrm{Grob}}$
  if $f = 0$ and $JP = \{\}$ then
    Return (true,$(0,0), B, N, W, \{\ \}, \{\ \}$)
  end if
  if $f \neq 0$ then
    $(cd, f, N, W)$ :=NewCond($f, N, W$)
    if $f = 0$ then
      Return (true,$(0,0), B, N, W, \{\ \}, \{\ \}$)
    end if
    if $cd \neq \{\ \}$ then
      $JP := JP \cup \{\ \mathrm{JPair}((u,f), r) \mid r \in R\}$
      add $(u,f)$ into $R$ and add $f$ into $B$
      sort $JP$ by increasing signature
    else
      Return (false,$(u,f), B, N, W, JP, R$)
    end if
  end if
```

Algorithm 5 Continuation of CondPGB

```
while $JP \neq \{\ \}$ do
    select and remove the first J-pair $P$ from $JP$
    $(v, g) :=$ Reduction$(P, R)$ performing only regular top reductions
    if $g = 0$ then
        $LM := LM \cup \{v\}$
        remove any pair in $JP$ if $v$ divides its signature
    else
        $g := \overline{g}^N$
        $(cd, g, N, W) :=$NewCond$(g, N, W)$
        if $cd = \{\ \}$ then
            if $g \neq 0$ then
                $JP := JP \cup \{\text{JPair}((v, g), r) \mid r \in R\}$
                add $(v, g)$ into $R$ and add $g$ into $B$
                sort $JP$ by increasing signature
            end if
        else
            if $g \neq 0$ then
                Return (false,$(v, g), B, N, W, JP, R$)
            end if
        end if
    end if
end while
Return (true,$(0, 0), B, N, W, \{\ \}, \{\ \}$)
```

Theorem 2. *Suppose that we are willing to compute a CGS for the ideal $\langle F \rangle$ with $F = \{f_{i+1}, \ldots, f_k\}$ for some i. Let N and W be the null and non-null conditions sets respectively and let σ a homomorphism such that (N, W) is its specification. Let also B be the set of polynomials satisfying the following conditions:*

- *$\sigma(B)$ is a basis of $\langle \sigma(F) \rangle$,*
- *$\sigma(\mathrm{LC}(g)) \neq 0$ for $g \in B$.*

Suppose that the CondPGB *algorithm outputs (test,B', N', W', JP', R'). If test = true, then $\sigma(B')$ is a Gröbner basis of $\langle \sigma(F) \rangle$, where (N', W') is a specification of σ. If test = false, then B' is an extended set of B*

which contains at least one polynomial for which the actual specification (N, W) cannot decide if its leading coefficient specializes to zero or not. The sets N, W, JP, R are also updated to N', W', JP', R' respectively.

Proof. The structure of the new CondPGB is similar to that of the old CondPGB in [18]. Therefore, the proof of this theorem may be deduced from [18, Proposition 15] and [8, Theorem 3.1]. $\square$

Remark 2. *We shall note that the finite termination of this algorithm is guaranteed by that of the GVW algorithm [8, Theorem 3.1].*

Remark 3. *In the above algorithm, if JP= $\emptyset$, it returns the sets B, N, W, where B is a Gröbner basis for the ideal $\langle f_{i+1}, \dots, f_k\rangle$ for some i w.r.t. the specification (N, W). To speed-up the computations, we can first minimize B; i.e. discard any polynomial in B whose leading monomial (w.r.t $\prec_{\mathbf{x}}$) is divisible by the leading monomial of another polynomial in B. Then, we may replace B by* `InterReduce`*$(B, \prec_{\mathbf{x},\mathbf{a}})$ to have the reduced Gröbner basis. Indeed, this may reduce the number of J-pairs for the next step.*

Below, we describe the CanSpec algorithm from [18] to produce k-quasi-canonical representation (N', W') for a specification (N, W).

Definition 5. *A specification (N, W) is called k-quasi-canonical if the following conditions hold:*

- *N is the reduced Gröbner basis of the ideal containing all polynomials that specialize to zero in $K[\mathbf{a}]$, w.r.t. $\prec_{\mathbf{a}}$.*
- *The polynomials in W specializing to non-zero are reduced modulo N and are irreducible over $K'[\mathbf{a}]$, where K' is an algebraic extension of K.*
- *$\prod_{q\in W} q \notin \sqrt{\langle N\rangle}$ and the polynomials in N are square-free.*
- *None of the polynomials in N have an irreducible factor contained in W.*

For more details we refer to [18].

Algorithm 6 CanSpec

```
Require: $\{\ N, W :$ where $(N, W)$ is an specification
Ensure: $\left\{ \begin{array}{l} test : \left\{ \begin{array}{ll} \text{true:} & \text{if } N \text{ and } W \text{ are compatible} \\ \text{false :} & \text{otherwise} \end{array} \right. \\ (N', W') : \text{a k-quasi-canonical representation of } (N, W). \end{array} \right.$
$W' :=$FacVar$(\{\overline{q}^N : q \in W\})$;
test:=true
$N' := N$ and $h := \prod_{q \in W} q$
if $h \in \sqrt{\langle N' \rangle}$ then
    test:=false
    $N' := \{1\}$
    Return (test , $(N', W')$)
end if
flag:=true
while flag do
    flag:=false
    $N''$:= Drop any factor of a polynomial in $N' \cap W'$, as well as multiple factors
    if $N'' \neq N'$ then
        flag:=true
        $N'$:= a Gröbner basis of $\langle N'' \rangle$ w.r.t. $\prec_{\mathbf{a}}$
        $W' :=$FacVar$(\{\overline{q}^{N''} : q \in W'\})$
    end if
end while
Return (test , $(N', W')$)
```

5 Experiments and results

We have implemented all the algorithms described in this paper in `Maple 15`[1]. In this section, we compare the performance of GVW-DisPGB algorithm with DisPGB and PGBMain algorithms. It should be noted that in PGBMain algorithm we use GVW algo-

[1]The Maple codes of the algorithms are available at `http://amirhashemi.iut.ac.ir/softwares`

rithm for the Gröbner basis computation. We use also the `cgsdr` function from `grobcov.lib` library which is a `Singular` [22] implementation of PGBMain. The following parametric ideals in the ring $S = K[a,b,c,d,m,n,r,t][x,y,z,u]$ were chosen, and our aim was to compute a CGS of each ideal w.r.t. the product of the orderings $u \prec_{lex} z \prec_{lex} y \prec_{lex} x$ and $t \prec_{lex} r \prec_{lex} n \prec_{lex} m \prec_{lex} d \prec_{lex} c \prec_{lex} b \prec_{lex} a$.

- EX.1 $= [x^3+(d-a)xy+m-a, -cba+az^2+cx-d, (c-a)y^2+xn+a, u^3+(a^2-1)x+n-m]$
- EX.2 $= [abu^4+b^2a^2+xyz-1, ay^2+n(mt-2)xz+a, baz^3+t(2-b^7)xyz+x^2z-1]$
- EX.3 $= [(c-1)y^3+(ac-b)x+dn, rx^5+(ba-c)z-n, z^3-(c-t)y]$
- EX.4 $= [(a+1)x^2+a^2b(d-1)y^2+a, y^2+bx+c(c^2-4), (a-d)z^3+ay^2+b(c^4-1)-1]$
- EX.5 $= [ax+by+cz^6, axy+byz^6+czx, xyz^6-1]$
- EX.6 $= [ax^2+cu+(a-3)x, bayx+ay^3-cz+1, (t-3)u^3+tu, bz^3+(1-b)z^2+mnx]$
- EX.7 $= [cz^5+dax^2+dby, yz^5b+axy+czx, dxyz^5-ba]$
- EX.8 $= [amy^2+x^3-zb-bt-1, amy^3+tx+n-t, z^3+tx-abcd]$
- EX.9 $= [(ba-c)z+rx^4-n, y^2+(ac-b)x+d, z^2-(cb-a)y+t]$
- EX.10 $= [(mn-a)x+y^2+a, cx+z^2, x^2+(ad-c)yx+ba, u^2+(a^4-4)x]$

Table 1. The performance comparison of different algorithms.

Example	Method	Time (Sec)	Used Memory (GB)	Branch
EX.1	GVWDisPGB	11.77	0.7	140
	DisPGB	26.59	1.65	149
	PGBMain	–	–	–
	`cgsdr`	–	–	–
	FirstGB	–	–	–
EX.2	GVWDisPGB	6.59	0.44	15
	DisPGB	33.45	2.21	14
	PGBMain	–	–	–
	`cgsdr`	–	–	–
	FirstGB	–	–	–
EX.3	GVWDisPGB	40.35	2.5	71
	DisPGB	72.81	4.85	59
	PGBMain	–	–	–
	`cgsdr`	–	–	–
	FirstGB	119.15	11.38	–
EX.4	GVWDisPGB	2.19	0.11	43
	DisPGB	–	–	–
	PGBMain	–	–	–
	`cgsdr`	–	–	–
	FirstGB	256.2	33.19	–
EX.5	GVWDisPGB	10.65	0.67	10
	DisPGB	–	–	–
	PGBMain	–	–	–
	`cgsdr`	–	–	–
	FirstGB	3.24	0.31	–
EX.6	GVWDisPGB	4.1	0.23	51
	DisPGB	82.11	4.66	36
	PGBMain	–	–	–
	`cgsdr`	0.5	–	24
	FirstGB	30.08	3.49	–
EX.7	GVWDisPGB	9.57	0.62	17
	DisPGB	–	–	–
	PGBMain	–	–	–
	`cgsdr`	–	–	–
	FirstGB	3.96	0.39	–

Continuation of Table 1.

Example	Method	Time (Sec)	Used Memory (GB)	Branch
EX.8	GVWDisPGB	102.3	5.85	43
	DisPGB	–	–	–
	PGBMain	–	–	–
	`cgsdr`	–	–	–
	FirstGB	5.40	0.55	–
EX.9	GVWDisPGB	10.38	6.7	119
	DisPGB	–	–	–
	PGBMain	–	–	–
	`cgsdr`	–	–	–
	FirstGB	–	–	–
EX.10	GVWDisPGB	22.38	1.4	245
	DisPGB	10.11	0.67	55
	PGBMain	–	–	–
	`cgsdr`	–	–	–
	FirstGB	136.8	12.45	–

We shall emphasize that from DisPGB we mean the classical DisPGB algorithm due to Montes [18]. The results are shown in the above tables, where the timings were conducted on a personal computer with 7 core, 8 GB RAM and 64 bits under the windows 7 operating system. All the computations are done over $\mathbb{Q}$. In these tables, the third and fourth columns show respectively the CPU time (in second) and the amount of required memory (in gigabytes) of the corresponding method computation. The last column indicates the number of branches of the output CGS. Furthermore, "First GB" method stands for the computation of the reduced Gröbner basis of the corresponding ideal in $K[\mathbf{a}, \mathbf{x}]$ w.r.t. $\prec_{\mathbf{x},\mathbf{a}}$ using the `Maple` function `Basis`. It is worth nothing that, this computation is needed in PGBMain to compute a CGS w.r.t. $\prec_{\mathbf{x},\mathbf{a}}$. Also, "–" means that the related function can not compute anything and we stopped the computation after 400 seconds. A comparison of the timing columns in the above tables and our test for some other examples show that this first implementation of GVWDisPGB is efficient for many examples.

According to our experiments on more than 50 examples, we may consider two main classes of examples for which the performance of

GVWDisPGB and PGBMain are quite different. In general, for each ideal I with $I \cap K[\mathbf{a}] \neq \langle 0 \rangle$ PGBMain has a better performance than GVWDisPGB. For instance, let us consider the ideal $I = \langle ax - ac + a^2, aby^3 + by + ab + b^2c, axy - by + ax - c, aby^3 + axy, by + bc + c^2 \rangle \subset K[a, b, c][x, y]$. By running both algorithms over I w.r.t. $c \prec_{lex} b \prec_{lex} a$ and $y \prec_{lex} x$, we observe that PGBMain takes about 20 seconds of CPU time while GVWDisPGB needs more than 95 seconds. It seems that when PGBMain finds a non-empty generating set for $I \cap K[\mathbf{a}]$, then this generating set has a positive impact on the rest of calculation. However, in contrast, if $I \cap K[\mathbf{a}] = \langle 0 \rangle$, then GVWDisPGB has a better performance than PGBMain. For instance, one can refer to EX.1 in Table 5.

The other issue concerning PGBMain is that by the structure of this algorithm, one needs to compute successive Gröbner bases by adding, at each step, new polynomials. This may enlarge the size of the computed Gröbner basis at each step. Let us consider the ideal $\langle aby^3 + b^2c + ab + by, axy + ax - by - c, aby^3 + axy \rangle$ in $K[a, b, c][x, y]$ with $c \prec_{lex} b \prec_{lex} a$ and $y \prec_{lex} x$. In Figure 5, the x-axis shows the number of computed branches and the y-axis indicates the number of polynomials as the input of Gröbner basis algorithm in each branch.

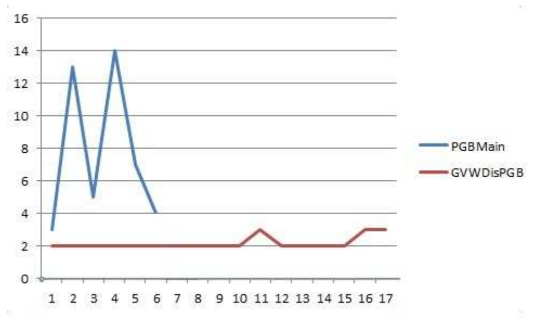

Figure 1. Comparing the output of GVWDisPGB and PGBMain

As it is shown, GVWDisPGB computes more branches than PGBMain (17 versus 6) but the maximum number of polynomials as the input of Gröbner basis algorithm in GVWDisPGB is 3 compared with

14 in PGBMain.

Let us consider the ideal $I = \langle f_1, f_2 \rangle \subset K[a,b][x,y]$ generated by the polynomials

$$\begin{aligned} f_1 &= (a^{12} + a^5 + 6)x^{17} + bx^5 + x^6 + x^4 + a^7x + a^5, \text{ and} \\ f_2 &= (a^{13} + a^6 + 3)y^7 + ay^3 + y + 1. \end{aligned}$$

One can observe that f_1 and f_2 have coprime leading monomials w.r.t $y \prec_{lex} x$ however this does not hold by considering f_1 and f_2 in $K[a,b,x,y]$ with the product of $b \prec_{lex} a$ and $y \prec_{lex} x$. Therefore, if the leading coefficients of f_1 and f_2 in $K[a,b][x,y]$ are non-zero, then $\{f_1, f_2\}$ forms a Gröbner basis by Buchberger's first criterion and in turn GVWDisPGB needs only 0.3 second to compute a CGS for I, however just the first Gröbner basis in $K[a,b,x,y]$ (which is the first step in PGBMain) takes more than five minutes.

Finally, it is worth noting that PGBMain has the advantage of using the Gröbner basis function of a computer algebra system. However, if the number of parameters is high then the first Gröbner basis computation may be hard and GVWDisPGB may have a better performance than PGBMain. Furthermore, due to the structure of PGBMain, the number of branches in the output CGS by using this algorithm is generally less compared with the one of GVWDisPGB.

Acknowledgements.

The authors would like to thank the anonymous reviewer for his/her valuable comments and suggestions to improve the quality of the paper.

References

[1] G. Ars and A. Hashemi, "Extended F$_5$ Criteria," *J. Symb. Comput.*, vol. 45, no. 12, pp. 1330–1340, 2010.

[2] T. Becker and V. Weispfenning, *Gröbner bases, a computational approach to commutative Algebra,* Springer, 1993, ISBN 978-1-4612-0913-3.

[3] B. Buchberger, "Ein Algorithms zum Auffinden der Basiselemente des Restklassenrings nach einem nuildimensionalen Polynomideal," PhD thesis, Universität Innsbruck, 1965.

[4] B. Buchberger, "A criterion for detecting unnecessary reductions in the construction of Gröbner bases", in *In symbolic and algebraic computation EUROSAM-1979*, (Marseille, France), Lecture Notes in Comput. Sci., vol. 72, Berlin: Springer, 1979, pp. 3–21.

[5] J.-C. Faugère, "A new efficient algorithm for computing Gröbner bases (F_4)," *J. Pure Appl. Algebra*, vol. 139, no. 1-3, pp. 61–88, 1999.

[6] J.-C. Faugère, "A new efficient algorithm for computing Gröbner bases without reduction to zero (F_5)", *International Symposium on Symbolic and Algebraic Computation ISSAC-2002*, Lille, France, 2002, ACM Press, pp. 75–83.

[7] S. Gao, Y. Guan and F. Volny IV, "A new incremental algorithm for computing Gröbner bases," in *International Symposium on Symbolic and Algebraic Computation ISSAC-2010*, (Munich, Germany), ACM Press, 2010, pp. 13–19.

[8] S. Gao, F. Volny IV and M. Wang, "A new framework for computing Gröbner bases," *Math. Comput.*, vol. 85, no. 297, pp. 449–465, 2016.

[9] R. Gebauer and H. Möller, "On an installation of Buchberger's algorithm," *J. Symb. Comput.*, vol. 6, no. 2-3, pp. 275–286, 1988.

[10] A. Hashemi and B. M.-Alizadeh, "Applying IsRewritten criterion on Buchberger algorithm," *Theoretical Computer Science*, vol. 412, no. 35, pp. 4592–4603, 2011.

[11] D. Kapur, Y. Sun and D. Wang, "A new algorithm for computing comprehensive Gröbner systems," in *International Symposium on Symbolic and Algebraic Computation ISSAC-2010*, (Munich, Germany), ACM Press, 2010, pp. 29–36.

[12] D. Kapur, Y. Sun and D. Wang, "An efficient algorithm for computing a comprehensive Gröbner system of a parametric polynomial system," *J. Symb. Comput.*, vol. 49, pp. 27–44, 2013.

[13] D. Kapur, Y. Sun and D. Wang, "An efficient method for computing comprehensive Gröbner bases," *J. Symb. Comput.*, vol. 52, pp. 124–142, 2013.

[14] D. Lazard, "Gröbner bases, Gaussian elimination and resolution of systems of algebraic equations," in *Proceedings of the European Computer Algebra Conference on Computer Algebra EUROCAL-1983*, (London, England), Lecture Notes in Comput. Sci., vol. 162, Berlin: Springer, 1983, pp. 146–156.

[15] M. Manubens and A. Montes, "Improving DisPGB algorithm using the discriminant ideal," *J. Symb. Comput.*, vol. 41, no. 11, pp. 1245–1263, 2006.

[16] M. Manubens and A. Montes, "Minimal canonical comprehensive Gröbner systems," *J. Symb. Comput.*, vol. 44, no. 5, pp. 463–478, 2009.

[17] H. M. Möller and F. Mora and C. Traverso, "Gröbner bases computation using syzygies," in *International Symposium on Symbolic and Algebraic Computation ISSAC-1992*, (Berkeley, USA), ACM Press, 1992, pp. 320–328.

[18] A. Montes, "A new algorithm for discussing Gröbner bases with parameters," *J. Symb. Comput.*, vol. 33, no. 1-2, pp. 183–208, 2002.

[19] A. Montes, "Solving the load flow problem using Gröbner bases," *ACM SIGSAM Bull.*, vol. 29, no. 1, pp. 1–13, 1995.

[20] A. Montes and M. Wibmer, "Gröbner bases for polynomial systems with parameters,"*J. Symb. Comput.*, vol. 45, no. 12, pp. 1391–1425, 2010.

[21] A. Suzuki and Y. Sato, “A simple algorithm to compute comprehensive Gröbner bases using Gröbner bases,” in *International Symposium on Symbolic and Algebraic Computation ISSAC-2006*, (Genoa, Italy), ACM Press, 2006, pp. 326–331.

[22] G.-M. Greuel, G. Pfister and H. Schönemann. SINGULAR *3.0. A computer algebra system for polynomial computations*, Centre for computer algebra, University of Kaiserslautern, 2011. [Online]. Available: http://www.singular.uni-kl.de.

[23] Singular grobcov.lib [Online]. Available: http://www-ma2.upc.edu/ montes/ (to download a beta version of the Singular grobcov.lib library).

[24] V. Weispfenning, “Comprehensive Gröbner bases,” *J. Symb. Comput.*, vol. 14, no. 1, pp. 1–29, 1992.

Amir Hashemi, Benyamin M.-Alizadeh,
Mahdi Dehghani Darmian

Received April 9, 2017

Amir Hashemi
Department of Mathematical Sciences, Isfahan University of Technology
Isfahan, 84156-83111, Iran
E–mail: `amir.hashemi@cc.iut.ac.ir`

Benyamin M.-Alizadeh
School of Mathematics and Computer Sciences, Damghan University
Damghan, 36716-41167, Iran
E–mail: `benyamin.m.alizadeh@gmail.com`

Mahdi Dehghani Darmian
School of Mathematics, Institute for Research in Fundamental Sciences (IPM)
Tehran, 19395-5746, Iran
E–mail: `m.dehghanidarmian@ipm.ir`