On Factorized Gröbner Bases

Hans-Gert Gräbe

Institut für Informatik
Universität Leipzig
Augustusplatz 10 - 11
04109 Leipzig (Germany)
graebe@informatik.uni-leipzig.de

Report No. 6

October 1994

Abstract

We report on some experience with a new version of the well known Gröbner algorithm with factorization and constraint inequalities, implemented in our REDUCE package CALI, [12]. We discuss some of its details and present run time comparisons with other existing implementations on well splitting examples.

Keywords : Gröbner bases with factorization, polynomial systems of equations, REDUCE, MAPLE, AXIOM.

1 Introduction

Let $S := k[x_1, \ldots, x_n]$ be the polynomial ring in the variables $x_1, \ldots, x_n$ over the field k and $B := \{f_1, \ldots, f_m\} \subset S$ a finite system of polynomials. Denote by $I(B)$ the ideal generated by these polynomials. One of the major tasks of constructive commutative algebra is the derivation of information about the structure of $Z(B) \subset \bar{k}^n$, the set of common zeroes of the system B over the algebraic closure $\bar{k}$ of k. For $C := \{g_1, \ldots, g_k\}$ denote the *relative set of zeroes* by

$$Z(B, C) := \{a \in \bar{k}^n : \forall\ f \in B\ f(a) = 0 \text{ and } \forall g \in C\ g(a) \neq 0\}.$$

The set C of inequalities can, of course, be replaced by a single inequality

$$\prod C := (\prod_{p \in C} p) \neq 0,$$

but for efficiency reasons we will keep them separated.

The introduction of such constraint conditions is not only motivated by our special aim, but also induced from applications, that often ask for solutions satisfying certain non degeneracy conditions as e.g. non vanishing determinants etc. Moreover, even starting with a "clean" problem, constraints come in during the Gröbner factorization algorithm in a natural way.

Splitting the system into smaller ones, solving them separately, and patching all solutions together is often a good guess for a quick solution of even highly nontrivial problems. As far as we know, such an approach was analysed first in greater detail by Czapor ([2], [3]), Davenport ([4]) and Melenk, Möller and Neun ([18], [19]). Of course, such a strategy makes sense only for problems that really will split, i.e. for reducible varieties of solutions. Problems coming from "real life" often fulfill this condition. So [18] presents an application of factorization and arithmetic restrictions to Gröbner bases of polynomial systems arising from large stationary chemical kinematics problems. This approach is also part of most of the general type Computer Algebra Systems. Below we discuss some new ideas included in the Gröbner factorizer implementation distributed with CALI, [12], and compare it with other existing implementations.

2 The Gröbner Algorithm with Factorization

As described in the introduction we consider the following

General Problem

> *Given a system $B = \{f_1, \ldots, f_m\} \subset S$ of polynomials and a set of side conditions C find a collection (B_α, C_α) of polynomial systems B_α in "triangular" form (here : being a Gröbner basis) and side conditions C_α such that*
>
> $$Z(B, C) = \bigcup_\alpha Z(B_\alpha, C_\alpha).$$

Using factorization this problem may be solved with the following well known algorithm, see e.g. [19] :

Factorized Gröbner Bases FGB(B,C)

- During a preprocessing interreduce B and try to factor each polynomial $f \in B$. If f factors, replace B by a set of new problems, one for each factor of f. Update the side conditions and apply the preprocessing recursively. This ends up with a list of interreduced problems with non factoring base elements.

- For each basis in the problem list compute its list of critical pairs and start the corresponding Gröbner basis calculations. Each such calculation consists of a polynomial list, a list of critical pairs not yet processed, and side conditions.

- Try each reduced (non zero) S-polynomial to factor before it will be added to the polynomial list. If it factors, split up the problem into as many subproblems as there are (different) factors, add each of the factors to the corresponding subproblem, and update the pair list and the side conditions.

- If the pair list is exhausted, extract the minimal Gröbner basis of the subproblem. If it is not yet interreduced (i.e. the reductum contains non standard terms), apply tail reduction to compute the minimal reduced Gröbner basis. This may cause some of the base elements to factor anew. Apply the preprocessing once more. If the result is stable then return it. Otherwise put the subproblems produced during the preprocessing back into the problem list.

Obviously this algorithm terminates and returns a list of Gröbner bases and constraints with the desired properties. Moreover, this approach may be parallelized in an easy master/slaves frame. In [14] we report on our experiences with such a parallelization based on the PVM-REDUCE version of Melenk and Neun.

Realizing the above general approach we use the following elementary operations :

1. **Updating after factorization**

   If (B, C) is a problem and $f \in I(B)$ factors as $f = g_1^{a_1} \dots g_m^{a_m}$ then replace the problem by the problem list

   $$\mathbf{NewCon}(B, C, \{g_1, \dots, g_m\}) := \{(B \cup \{g_i\}, C \cup \{g_1, \dots, g_{i-1}\}) \mid i = 1, \dots, m\}$$

2. **Inconsistency check**

   (B, C) is inconsistent, i.e. $Z(B, C) = \emptyset$, if the normal form $NF(c, B) = 0$ for some $c \in C$.

3. **Subproblem removal check**

   (B_1, C_1) can be removed if there is a problem (or partial result) (B_2, C_2) such that $Z(B_1, C_1) \subset Z(B_2)$. This occurs if $NF(f, B_1) = 0$ for all $f \in B_2$. The second problem has to be replaced by $(B_2, C_1 \cap C_2)$.

Both checks use not the full power of information but only sufficient conditions. Indeed, the side condition $C_1 \cap C_2$ in (3) is weaker than the (logical) disjunction of C_1 and C_2. But the latter may not correspond to a main open set in the Zariski topology. Since $C_1 \supset C$ and $C_2 \supset C$ we obtain nevertheless $C_1 \cap C_2 \supset C$. Hence the total set of solutions is not enlarged. For (2), we remark that e.g. the full inconsistency check would need a radical membership test, i.e. another (full) Gröbner basis calculation. This is impossible in the given frame since all checks must be easy enough not to influence the performance of the main algorithm to heavily. If B is the *Gröbner* basis of a *prime* ideal, $NF(c, B) = 0$ for some $c \in C$ is also necessary for $Z(B, C) = \emptyset$. Since our connection of factorization and Gröbner basis computation leads often to such bases, one should force to progress with a subproblem as deep as possible to take best advantage of the side conditions (and the removal check).

Splitting problems recursively in sets of subproblems yields a tree structure as described in [18] and conceptually used in both the REDUCE and the AXIOM implementations. With the desired parallelization in mind we will consider a master/slave frame instead. A master manages the set of problems and the set of results and distributes subtasks to one (or several) slaves to be treated by them. This has the following advantages :

- The data structures are one-dimensional lists of problems resp. results. Such data structures are best suited for the management by LISP-like languages.
- One can easily keep them sorted by appropriate sort criteria, this way forcing special selection strategies for the next problem(s) to be sent to the slave(s). Especially, splitting one of the leaves of the tree into subproblems one can queue up *all* these problems and continue with a different subproblem. Compared to the recursive approach as e.g. implemented in AXIOM this may lead to a significant speedup, although by the above theoretical remark and empirical observations a depth first strategy has to be preferred.
- One can easily apply the subproblem removal check not only to the current problem, but to all problems (and results) queued up. This may lead to significant savings cancelling unnecessary branches in an early stage of the computation.

As explained above we should force a problem sort strategy that mimics a depth first recursion on the corresponding tree. We approximate this strategy sorting the problems by their virtual dimension. This is the dimension of the current lead term ideal $lt(B)$. Computing new S-polynomials the lead term ideal grows and hence its dimension eventually drops. Thus our strategy forces the slaves to treat partial Gröbner bases in greatest progress first.[1] Updating the global problem list with newly produced subproblems the master applies the removal check on both the new and the old problems (and results).

For details of the implementation we refer the reader to the source code of CALI, available through [12].

[1] This is not the whole story since we use the easy linear time dimension algorithm, that works properly only for unmixed ideals, see [13]

3 The Preprocessing

We organized the preprocessing in a recursive way factoring each time a single basis element and then interreducing and updating the corresponding subproblems before the preprocessor is called on them anew. This has some advantage against the complete factorization of *all* base elements in one step. To see this we restrict our considerations to ideals generated by monomials, this way discussing the influence of common factors occuring in several base elements on the preprocessing, but not the tail reduction effects derived from them.

To compare our recursive approach with a complete factorization of all base elements, note that e.g. Reisner's example, [21], an ideal generated by 10 square-free monomials of degree 3, would split into 3^{10} subproblems with the latter approach, whereas it splits into 10 prime monomial ideals with 46 intermediate subproblems with the former one. This is true for almost all examples containing many splitting basis elements. The following general example illustrates the situation once more :

EXAMPLE :

$$I_{m,n} := I(x_{2k-1}x_{2l} : 1 \le k \le m, 1 \le l \le n)$$

As easily seen, $I_{m,n} = I(x_{2k-1} : 1 \le k \le m) \cap I(x_{2l} : 1 \le l \le n)$ decomposes finally into two subproblems. Factoring all the mn generators at once and combining them to subsystems yields 2^{mn} different ideals (of coordinate hyper-spaces). Two of them are the above minimal ones with respect to inclusion.

Using a recursive splitting argument we produce $I(x_2, x_4, \ldots, x_{2n})$ on the main branch and successively $I_0, I_1, \ldots, I_{n-1}$ in the problem list with

$$I_0 = I(x_{2m-1}) + I_{m-1,n}$$

and

$$I_k = I(x_{2m-1}, x_{2n}, \ldots, x_{2(n-k)}) + I_{m-1,n-k} \qquad \text{for } k > 0.$$

Since $I_0 \subset \ldots \subset I_{n-1}$ only one problem survives and $I_{m,n}$ splits recursively generating only mn intermediate subproblems.

4 Comparison with other Gröbner factorizer implementations

Like the original Buchberger algorithm the Gröbner factorization method may be combined with different term orders. As a main ingredient of the polynomial systems' solver of almost all major general purpose Computer Algebra Systems its combination with a pure lexicographic term order plays an important role. In such a setting the resulting Gröbner bases are often in a "triangular form" convenient for further processing, see e.g. [20, ch. 4].

Although (or better : since) factorized Gröbner bases with respect to a pure lex. term order carry important information usually they are hard to compute. Intermediate coefficient swell and high degrees in the output polynomials are common (and upto now not well understood) phenomena to be expected in advanced applications. Hence FGB computations with respect to "cheaper" term orders (especially the degrevlex one) should be involved to extract at least some information about the underlying polynomial system, if a direct attack via a pure lex. term order does not succeed.

Below we collected some examples known nowadays as benchmark tests for the original Buchberger algorithm and applied to them the FGB algorithm combined with the pure lex. term order with respect to the given list of variables (in decreasing order) as far as it was possible, then switching to the degrevlex term order.

The examples are the following :

G1 : [8, eq. (4)], see also [9, ex. 1], [11, 3.1] or [1].

G6 : [8, eq. (8)], see also [9, ex. 2].

G7 : [7, eq. (6)], see also [9, ex. 3] (note that the system is homogeneous).

K4 : The Katsura example with 4 variables, [1].

A5 : The cyclic roots example with 5 variables, [11, 3.2].

Go : The (quasi)homogenized version of Gonnet's example from [1]
(with $deg\ a_i = deg\ b_i = 1, deg\ c_i = 2$).

S_k : Schwarz' examples (communicated to us by G. Pfister) :

$$vars := \{x_1, \ldots, x_k\}$$

$$\begin{aligned} s_5 := \{\ & x_1^2 + x_1 + 2x_2x_5 + 2x_3x_4\,, \\ & 2x_1x_2 + x_2 + 2x_3x_5 + x_4^2\,, \\ & 2x_1x_3 + x_2^2 + x_3 + 2x_4x_5\,, \\ & 2x_1x_4 + 2x_2x_3 + x_4 + x_5^2\,, \\ & 2x_1x_5 + 2x_2x_4 + x_3^2 + x_5\} \end{aligned}$$

$$\begin{aligned} s_6 := \{\ & x_1^2 + x_1 + 2x_2x_6 + 2x_3x_5 + x_4^2\,, \\ & 2x_1x_2 + x_2 + 2x_3x_6 + 2x_4x_5\,, \\ & 2x_1x_3 + x_2^2 + x_3 + 2x_4x_6 + x_5^2\,, \\ & 2x_1x_4 + 2x_2x_3 + x_4 + 2x_5x_6\,, \\ & 2x_1x_5 + 2x_2x_4 + x_3^2 + x_5 + x_6^2\,, \\ & 2x_1x_6 + 2x_2x_5 + 2x_3x_4 + x_6\} \end{aligned}$$

$$\begin{aligned} s_7 := \{ & x_1^2 + x_1 + 2x_2x_7 + 2x_3x_6 + 2x_4x_5\,, \\ & 2x_1x_2 + x_2 + 2x_3x_7 + 2x_4x_6 + x_5^2\,, \\ & 2x_1x_3 + x_2^2 + x_3 + 2x_4x_7 + 2x_5x_6\,, \\ & 2x_1x_4 + 2x_2x_3 + x_4 + 2x_5x_7 + x_6^2\,, \\ & 2x_1x_5 + 2x_2x_4 + x_3^2 + x_5 + 2x_6x_7\,, \\ & 2x_1x_6 + 2x_2x_5 + 2x_3x_4 + x_6 + x_7^2\,, \\ & 2x_1x_7 + 2x_2x_6 + 2x_3x_5 + x_4^2 + x_7\} \end{aligned}$$

$$
\begin{aligned}
s_8 := \{&x_1^2 + x_1 + 2x_2x_8 + 2x_3x_7 + 2x_4x_6 + x_5^2, \\
&2x_1x_2 + x_2 + 2x_3x_8 + 2x_4x_7 + 2x_5x_6 , \\
&2x_1x_3 + x_2^2 + x_3 + 2x_4x_8 + 2x_5x_7 + x_6^2, \\
&2x_1x_4 + 2x_2x_3 + x_4 + 2x_5x_8 + 2x_6x_7 , \\
&2x_1x_5 + 2x_2x_4 + x_3^2 + x_5 + 2x_6x_8 + x_7^2, \\
&2x_1x_6 + 2x_2x_5 + 2x_3x_4 + x_6 + 2x_7x_8 , \\
&2x_1x_7 + 2x_2x_6 + 2x_3x_5 + x_4^2 + x_7 + x_8^2, \\
&2x_1x_8 + 2x_2x_7 + 2x_3x_6 + 2x_4x_5 + x_8\}
\end{aligned}
$$

We compared our implementation with other existing ones (REDUCE 3.4.1, AXIOM 1.0, MAPLE V.2) on an IBM/RS 6000.[2] In table 1 we collected the results of these experiments with the several Gröbner factorizer implementations. The first column contains the corresponding computation (CPU-)time in sec. as reported from the system, the second the number # SP of final subproblems returned by the corresponding Gröbner factorizer. Since AXIOM and MAPLE produce in general subsystem lists that are not reduced with respect to subideal relationship we report both the number of subproblems returned by the system and the number of essential subproblems among them.

Table 1 is divided into three parts. The first part contains easy examples, the second and third part more difficult ones. The first two parts are computed with respect to the pure lexicographic term order, the latter with respect to the degree-wise reverse lexicographic term order (*gsolve* in our MAPLE version can be combined only with plex, so no comparison was possible in the latter examples).

[2] MATHEMATICA 2.1. doesn't offer explicit access to a Gröbner factorizer. Its SOLVE-function invokes a certain factorization strategy, but without user's influence to choose the underlying term order. For the lex. examples we've got the following behavior (on an HP 735) :

- A5 and S5 it was unable to crack.
- For G1 it reports after 1025 s. a list of about 10000 solutions with many repetitions of dimension ≤ 1, that we did not try to analyze.
- For K4 it reports after 1.0 s. the two linear solutions and another one with nested roots of multiplicity 12. In the original ideal (being already radical) this component has degree 6.
- For Gonnet's example it reports after 58.2 s. 20 solutions, all of dimension 4, containing only two really different ones (The ideal has dimension 7).
- The same applies to G6 : After 23.6 s. there were returned 6 one-dimensional solutions, missing $\{\lambda_3 = \lambda_4 = 0\}$ and $\{\lambda_4 = \lambda_5 = 0, \lambda_1 = 1\}$.

ex.	CALI		REDUCE		AXIOM		MAPLE	
	time	# SP	time	# SP	time	# SP	time	# SP
G1	3.5	9	1.7	9	7.5	16/9	32.7	22/12
G6	1.2	8	0.75	8	13.5	12/8	7.9	13/11
K4	1.8	3	1.6	3	6.8	3	6.7	3
A5	16.1	15	7.9	15	45.4	16/15	9730	18/13
Go	15.3[3]	7	46.0	9[4]	2022	192/7	430[5]	32…40/7
S5	80.1	12	21.8	12	39.6	13/12	233	14/12
S6	> 90000		> 90000		182	35/32	> 38000	
G7	211	20	15.7	20	1350	266/22	> 17000	
S5	1.8	6	0.9	6	11.4	6	-	-
S6	7.7	22	4.6	20	81.1	21/20	-	-
G7	1092	21	3428	21	2281	200/20	-	-
S7	32.0	8	24.3	8	267	9/8	-	-
S8	710	63	3073	59	2190	64/63	-	-

Table 1 : Run time experiments with different Gröbner factorizers

Let's add some remarks about the quality of the output beyond CPU time. The occurrence of superfluous (embedded) solutions in both the MAPLE and AXIOM outputs is due to the recursive implementation suggested by the inherent tree structure and self-similarity of the algorithm. The elimination of such subproblems by the user later on, although easy from an algorithmic point of view, is difficult in practise, since for the necessary subideal test one has to go into deep system's details.

On the examples of part 1 the MAPLE answer differs from the other ones (that are nearly optimal in the sense explained below). This is due to an inaccurate implementation that often returns not completely factorized solutions. E.g. the output of G1 contains 3 components with base elements λ_1^2 and λ_1^4 among the (factorized) generators, whereas G6 gives solutions containing $\lambda_3^2(\lambda_1-1), \lambda_4^3$ etc. Another surprising fact we observed with Go : The (conceptually deterministic) serial implementation led in different runs to solution lists of different length !

Solving systems of polynomial equations in an ultimate way means to find the isolated primes of the associated variety and to present them in a way that is well suited for further computations. Good presentations of prime ideals are e.g. a regular (polynomial) parametrization (reg. par.)

$$B = \{x_k - p_k(x_1,\ldots,x_d) \mid k = d+1,\ldots,n\},\ p_k \in k[x_1,\ldots,x_d]$$

or a zero-dimensional prime in general position (g.p.) [10, prop. 7.1]

$$B = \{x_k - p_k(x_n) \mid k = 1,\ldots,n-1\} \cup \{p_n(x_n)\},\ p_k \in k[x_n].$$

Generalizations include zero-dimensional triangular sets ([15], strong triangular sets in [20, ch. 4]), different generalizations to positive dimension ([16], [22]), rational parametrizations,

[3] With a better ecart (giving c_i weights 2) only 8.3 s.
[4] Two of them are not Gröbner bases due to a code bug.
[5] average value

characteristic sets etc. For a full decomposition into prime components the only known general algorithmic approach is that of [10].

Since the FGB algorithm is a heuristic approach, one cannot expect to get such a full decomposition. Nevertheless for small examples and a pure lexicographic term order, it often ends up with a list of primes in a convenient presentation. This is very important, since it is not easy to extract all solutions of a system of polynomial equations even from a pure lex. Gröbner basis. E. g. in [8] the authors found for both G1 and G6 not all solutions : For G1 they missed the two-dimensional component

$$\{\lambda_1 = \lambda_2 = \lambda_4 = \lambda_5 = \lambda_7 = 0\}$$

and two one-dimensional parts and for G6 the one-dimensional components

$$\{\lambda_1 = \lambda_3 = \lambda_5 = 0\} \quad \text{and} \quad \{\lambda_1 = \frac{1}{3}, \lambda_3 = 0, \lambda_4 = -\lambda_5\}.$$

In table 2 we collected some of the output characteristics of our sample computations.

example	# SP	Dimensions	Structure
G1	9	1x3 3x2 5x1	all reg. par.
G6	8	1x2 7x1	all reg. par.
K4	3	3x0	primes, all in g.p.
A5	15	15x0	all strong triangular, but 20 primes
Go	7	1x7 1x6 2x5 3x4	all prime, but of difficult structure
S5	12	12x0	primes, all in g.p.
S6	32	32x0	28 primes in g.p. and 4 strong triangular systems, decomposing into (4 4 2 2) comp. 40 primes in total
G7	20	4x6 4x5 12x4	see below
S5	6	6x0	2 primes and 4 subsystems, decomposing into (3 3 2 2) comp.
S6	22	22x0	6 primes and 16 subsystems, decomposing into (2x3 14x2) comp.
G7	21	3x6 4x5 12x4 2x3	see below
S7	8	8x0	not easy, 28 primes
S8	63	63x0	not easy, 96 primes

Table 2 : Output characteristics

For larger examples and especially with respect to the degrevlex term order output subproblems may consist of several prime components, which the system was not able to split. Due to different pair selection strategies this may apply to some or all of the implementations thus explaining differing numbers of subsolutions.

The relations between different outputs may be even more difficult. For the G7 example e.g. we obtained the following characteristics :

> The system decomposes minimally into 20 primes of dimension (4x6 4x5 11x4 1x3).

The (quick) pure lex. decomposition of both CALI and REDUCE yields 17 prime components with quite difficult structure and 3 four-dimensional subsystems of degree 4, 5, 5. CALI's *isolatedprimes* decomposes the latter two subsystems into four-dimensional primes of degree 4 and 1 each, whereas the former decomposes into a four-dimensional component of degree 4 and the three-dimensional component of degree 6. In a second minimization step some of the old and some of the new four-dimensional components turn out to be superfluous for a minimal decomposition.

AXIOM's output for the pure lex. term order contains 20 primes and two composite subsystems. One of them, of dimension 2, contains a generator a_0^2 (?) and is radically embedded. The other composite subsystem is of dimension 4 and decomposes into three components of dimension (4 3 3). Again some of the old and some of the new components turn out to be superfluous.

For the degrevlex term order CALI's decomposition contains 3 composite subsystems that split into components of dimension (6 6), (3 3) and (3 4).

REDUCE's decomposition contains 4 composite subsystems with almost the same properties, whereas

AXIOM produces 17 primes and 3 composite systems (of dimension (6 4 3)).

Note that Gerdt *et al.* report in [9] with ASYS (and another method) that they obtained a complete list of 76 subsolutions.

A good guess for examples with many composite subsystems in the output collection, arising e.g. in combination with the degrevlex term order, is a pure lexicographic post-processing of the results obtained so far. For this purpose one has to interreduce the base polynomials in the output list with respect to a pure lexicographic term order and to restart another FGB computation for each of them. One can apply the same shortcuts as discussed above to the whole list of these problems instead of processing each of the problems individually. Table 3 contains the results of the corresponding computations in our experimental implementation for the examples of part 3 in table 1.

example	FGB-D	# SP	Inter	FGB-L	# SP	Structure
S5	1.8	6	0.6	1.9	12	all prime and in g.p.
S6	7.7	22	0.3	5.7	38	2 of them are composite
G7[6]	1092	21	0.2	2.9	22	all but 2 are prime
S7	32.0	8	> 2900[7]			see below
S8	710	63	10.7	46.0	90	6 of them are composite

Table 3 : Combining degrevlex. and lex. FGB computations

Here

FGB-D denotes the CPU time (in sec.) for the degrevlex FGB computation,

Inter denotes the time spent for the interreduction of these results wrt. a pure lexicographic term order, and

[6]Since the system is homogeneous, this is merely a turn from degrevlex to deglex.
[7]Heap space low.

FGB-L denotes the time spent for the lex. FGB computation on the list of the interreduced results.

 # SP counts the number of subproblems, into which the original problem splitted so far

and the last column reports about the quality of the final result obtained this way.

Some words about S7 : The 8 subsystems, produced by the degrevlex term order are of degree (1 1 7 7 21 21 35 35). The former 6 subsystems decompose with the approach discussed so far into 14 primes (in g.p.) of degree 1, 3 and 6. The really hard part are the components of degree 35. We even failed to interreduce these components wrt. the pure lexicographic term order. Using the FGLM linear algebra approach of [5] with precomputed borderbasis as described in [17] the degrevlex Gröbner bases of these components can be converted into the lexicographic ones in 6.5 s. each. Since both are in g.p. another application of the Gröbner factorizer yields their prime decompositions. This way we obtain 2x7 new prime components. Since none of them turns out to be superfluous, the polynomial system S7 decomposes finally into 28 zerodimensional primes.

This approach is in general a good additional guess for zero-dimensional ideals. It allows to find solutions of systems that are otherwise untractable (without special tricks). So e.g. for the Caprasse/Demaret examples in dimension $4\ldots 6$, $C4\ldots C6$, see [6], we get even without the special factoring elements constructed in [6] the following results :

example	FGB-D	# SP	Change	FGB-L	# SP	Structure
C4	23.6	8	2.2	2.3	13	all prime and in g.p.
C5	196	19	116	17.4	35	all but 2 are prime
C6	27160	44	3953	186	109	all but 5 are prime of degree $1\ldots 12$.

Table 4 : Combining degrevlex. FGB, FGLM, and lex. FGB computations

Here, in addition to the notion introduced above, *Change* denotes the time spent for the FGLM term order change.

We conclude, that these approaches are good candidates to be tried, if a direct lexicographic FGB attack fails. Of course, it makes sense only for such problems that really admit a factorization with respect to a "cheaper" term order. Since these pieces are of smaller size than the original problem, the lexicographic FGB algorithm hopefully goes through on them.

In a forthcoming paper we will discuss less obvious strategies to split problems, where components keep glueing together.

References

[1] W. Boege, R. Gebauer, H. Kredel : Some examples for solving systems of algebraic equations by calculating Gröbner bases. *J. Symb. Comp.* **2** (1986), 83 - 98.

[2] S. R. Czapor : Solving algebraic equations via Buchberger's algorithm. In : Proc. EUROCAL'87, LNCS 378 (1987), 260 - 269.

[3] S. R. Czapor : Solving algebraic equations : Combining Buchberger's algorithm with multivariate factorization. *J. Symb. Comp.* **7** (1989), 49 - 53.

[4] J. H. Davenport : Looking at a set of equations. Bath Comp. Sci. Technical Report 87-06 (1987).

[5] Faugere, Gianni, Lazard, Mora : Efficient computations of zerodimensional Gröbner bases by change of ordering. Technical Report, 1989. To appear in *J. Symb. Comp.*.

[6] K. Gatermann : Symbolic solution of polynomial equation systems with symmetry. In : Proc. ISSAC'90, Addison Wesley 1990, 112 - 119.

[7] V. P. Gerdt, A. Yu. Zharkov : Computer classification of integrable coupled KdV-like systems. *J. Symb. Comp.* **10** (1990), 203 - 207.

[8] V. P. Gerdt, N. V. Khutornoy, A. Yu. Zharkov : Solving algebraic systems which arise as necessary integrability conditions for polynomial nonlinear evolution equations. In : Computer algebra in physical research (ed. Shirkov, Rostovtsev, Gerdt), World Scientific, Singapore 1991, 321 - 328.

[9] V. P. Gerdt, N. V. Khutornoy, A. Yu. Zharkov : Gröbner basis technique, homogeneity, and solving polynomial equations. Preprint, JINR E11-92-157, Dubna 1992.

[10] P. Gianni, B. Trager, G. Zacharias : Gröbner bases and primary decomposition of polynomial ideals. *J. Symb. Comp.* **6** (1988), 149 - 167.

[11] A. Giovini *et al.* : "One sugar cube, please" or selection strategies in the Buchberger algorithm. In Proc. ISSAC'91, ACM Press, 1991, 49 - 54.

[12] H.-G. Gräbe : CALI – A REDUCE package for commutative algebra. Version 2.1., Oct. 1993. Available through the REDUCE library e.g. at `redlib@rand.org`.

[13] H.-G. Gräbe : Two remarks on independent set. *J. Alg. Comb.* **2** (1993), 137 - 145.

[14] H.-G. Gräbe, W. Lassner : A parallel Gröbner factorizer. In : Proc. PASCO'94 Linz, World Scientific, Singapore 1994, 48 - 53.

[15] D. Lazard : Solving zerodimensional algebraic systems. *J. Symb. Comp.* **13** (1992), 117 - 131.

[16] D. Lazard : A new method for solving algebraic systems of positive dimension. *Discr. Appl. Math.* **33** (1991), 147 - 160.

[17] M. Marinari, H.-M. Möller, T. Mora : Gröbner bases of ideals given by dual bases. In : Proc. ISSAC'91, ACM Press 1991, 55 - 63.

[18] H. Melenk, H.-M. Möller, W. Neun : Symbolic solution of large chemical kinematics problems. *Impact of Computing in Science and Engineering* **1** (1989), 138 - 167.

[19] H. Melenk : Practical applications of Gröbner bases for the solution of polynomial equation systems. In : Computer algebra in physical research (ed. Shirkov, Rostovtsev, Gerdt), World Scientific, Singapore 1991, 230 - 235.

[20] B. Mishra : Algorithmic Algebra. Springer, New York 1993.

[21] G. Reisner : Cohen Macaulay quotients of polynomial rings. *Adv. math.* **21** (1976), 30 - 49.

[22] D. Wang : An elimination method for solving polynomial systems. *J. Symb. Comp.* **16** (1993), 83 - 114.