A "Divide and Conquer" Algorithm for Hilbert-Poincaré Series, Multiplicity and Dimension of Monomial Ideals*

Anna Maria Bigatti1 Pasqualina Conti2 Lorenzo Robbiano1 Carlo Traverso3

November 10, 1993

1 Introduction

In the computation of the Hilbert-Poincaré series of homogeneous ideals, the known algorithms, [MM], [KP], [BS], [BCR] have a first algebraic step coinciding with the computation of the associated Gröbner basis w.r.t. any ordering and the corresponding initial ideal (the *associated staircase*), and a second combinatorial step that from the staircase computes the Hilbert-Poincaré series.

A well-known classical algorithm computes the Hilbert-Poincaré series from a free resolution, but is practically infeasible because of its complexity. The algorithms of [MM] use an inclusion-exclusion counting technique; the algorithms of [KP] and [BCR] proceed by induction on the dimension; the algorithm of [BS] proceeds by induction on the number of generators of the initial ideal (the cogenerators of the staircase).

Usually, combinatorial algorithms can be speeded up by a "Divide and Conquer" approach: splitting the problem into two smaller problems of ap-

*This research was performed with the contribution of C.N.R., M.U.R.S.T, and CEC contract ESPRIT B.R.A. n.6846 POSSO.

E-mail addresses: bigatti@unimat.to.cnr.it, conti@dm.unipi.it, robbiano@igecuniv.bitnet, traverso@dm.unipi.it.

proximately the same size. In successful cases this trades a linear step for a logarithmic step, and can reduce from exponential to polynomial complexity.

Our approach explains how to split a staircase through the choice of a monomial (the *pivot*), then we discuss how to design a strategy for the choice of the pivot. The worst case complexity is not improved, since in some extreme cases every splitting is bad, (the computation of Hilbert-Poincaré series is at least as difficult as a NP-complete problem in the number of variables, see [BS]) but in several practical cases the situation is much better; in particular, our algorithm in the best case has a complexity that is a linear factor better than the best case of [BS], and can be specialized, with a choice of the splitting strategy, to the algorithm of [BCR]. In practice, a simple random strategy is quite good, avoids the costly computations involved in choice of an optimal variable of [BCR], and marginally improves the performance even in the optimal Borel-normed case.

The same approach allows to compute the dimension and the multiplicity. If only the dimension is needed, a standard improvement (computing the radical) allows a speeding of the algorithm, and the knowledge of the dimension allows to simplify the finding of the multiplicity.

The algorithms have been implemented, both in CoCoA, [GN] and AlPi, [TD]. Some test cases are given.

2 Staircases

A *staircase* S (also called *Ferrer diagram* or *order ideal of monomials*) is a set of elements of $\mathbf{N}^n$ such that if $(a_1, \ldots, a_n) \in S$ and $b_i \leq a_i$ then $(b_1, \ldots, b_n) \in S$.

On $\mathbf{N}^n$ there is a partial ordering $(a_1, \ldots, a_n) \leq (b_1, \ldots, b_n)$ iff $a_i \leq b_i$ for each i. The corresponding lattice operations $\alpha \wedge \beta$, $\alpha \vee \beta$ are the componentwise min and max.

Elements of $\mathbf{N}^n$ correspond to power products (terms, monic monomials) of $k[x_1, \ldots, x_n]$; the partial ordering corresponds to divisibility ($\alpha \leq \beta \iff X^\alpha \mid X^\beta$). We often identify α with X^α, and use notations referring to both indifferently. In particular, if $\alpha = (a_1, \ldots, a_n) \in \mathbf{N}^n$, $|\alpha| = \sum a_i$ will be called the *degree* of α. The operations $\wedge$ and $\vee$ correspond to GCD and lcm.

We say that an element of $\mathbf{N}^n$ is a *pure power* if it has only one coordinate that is non zero. Otherwise it is called *mixed*.

If $I \subseteq k[x_1, \ldots, x_n]$ is an ideal, and we have an ordering, then the staircase associated to I is the set of exponents $\alpha = (a_1, \ldots, a_n)$ such that no element of I has leading power product equal to X^α. The staircase associated to I can be computed through a Gröbner basis, taking all exponents such that no leading power product of the Gröbner basis divides them, and the X^α constitute a linear basis of $k[X]/I$. Conversely, from the staircase we can recover the leading power products of the reduced Gröbner basis, corresponding to the minimal elements of the complement of the staircase.

The staircase is usually given through these minimal elements, that are called the *minimal cogenerators* of the staircase. Given a set G of elements of $\mathbf{N}^n$, there is a maximal staircase disjoint from G, and it is called the staircase *cogenerated* by G. It is the complementary of the monoideal generated by G.

We denote with $[\alpha_1, \ldots, \alpha_m]$ the staircase cogenerated by $\alpha_1, \ldots, \alpha_m$; if S is a staircase, denote with $[S]$ the minimal set of cogenerators of S. In particular, $[[\alpha_1, \ldots, \alpha_m]]$ is obtained deleting from $\{\alpha_1, \ldots, \alpha_m\}$ all the elements that are multiple of another element (the notation implicitly assumes that no duplications appear in $\{\alpha_1, \ldots, \alpha_m\}$). The algorithm for operating such deletions is an essential tool, and its efficient implementation is very important. The issue is discussed in section 6.

Given two staircases S_1, S_2, we say that S_1 is *strictly smaller* than S_2 if S_1 is a proper subset of S_2 and moreover an injective map ϕ exists from $[S_1]$ to $[S_2]$ such that $\phi(\alpha) \geq \alpha$. Given a staircase, only a finite number of strictly smaller staircases exists.

A *T-staircase* is a translate of a staircase; all that is said for staircases applies, with minor modifications, to T-staircases. Most of what we will prove will be applicable to T-staircases without modifications, and we will not even quote it.

Given a staircase (or a T-staircase) S one defines its Hilbert-Poincaré series being the formal power series in the indeterminate T defined by $\sum_{i=0}^{\infty} d_i T^i$, where d_i is the number of elements of S of degree i, and is denoted by H_S (of course, this being an infinite formula, it is not an algorithm). Clearly, if $S' = \alpha + S$, then $H_{S'} = T^{|\alpha|} H_S$.

If I is a homogeneous ideal of $k[X]$, the Hilbert-Poincaré series of $k[X]/I$ is defined, and coincides with the Hilbert-Poincaré series of the associated staircase. This is the motivation of the interest in computing the Hilbert-Poincaré series of staircases.

Sometimes, one is not interested in the Hilbert-Poincaré series, but only in the *dimension* and the *multiplicity* of the staircase. These are defined as follows. Consider a staircase S, and let $\sum_0^{\infty} d_i T^i$ be its Hilbert-Poincaré series. Define $\phi(r) = \sum_0^r d_i$. Then a well-known theorem states that $\phi(r) = \frac{e}{d!} r^d + O(r^{d-1})$, for suitable integers $e > 0$, $d \geq 0$. Then e is the multiplicity and d is the dimension, and the dimension is smaller than or equal to the number n of variables. The difference $n - d$ is called *codimension*. We denote by e_S, δ_S, c_S the multiplicity, dimension and codimension of S.

Moreover, $H_S = p(T)/(1-T)^d$, where $p(T)$ is a polynomial such that $p(1) = e_S$, $d = \delta_S$. We denote by $\langle S \rangle$ the polynomial $(1-T)^n H_S$, where n is the number of variables; it has a zero in 1 of order $n-d$. We write $\langle \alpha_1, \ldots, \alpha_m \rangle$ for $\langle [\alpha_1, \ldots, \alpha_m] \rangle$; the aim of the algorithms is to compute $\langle \alpha_1, \ldots, \alpha_m \rangle$ from $(\alpha_1, \ldots, \alpha_m)$.

The description of the algorithm and its correctness do not depend on the unproved assertions stated above, and the algorithm itself is a proof of them.

Everything that will be proved in this paper applies to staircases of modules without changes; a staircase associated to a submodule of a free module is a disjoint union of T-staircases, hence the Hilbert-Poincaré series is the sum of separately computable Hilbert-Poincaré series of T-staircases. Of course, some information can be present relying the different T-staircases, and this could be used to speed up the algorithms, but in general this information is not available. This issue needs further study and experimenting.

3 Splitting a Staircase

We will consider two types of splitting of staircases: as a product of staircases (a *vertical splitting*), and as disjoint union of a staircase and a T-staircase (a *horizontal splitting*).

A vertical splitting is possible if and only if we can identify two disjoint subsets X_1 and X_2 of the variables such that any minimal cogenerator is a power product in either the variables X_1 or X_2. In that case, S is a product of two staircases S_1 and S_2, in X_1 and X_2, each one cogenerated by the corresponding cogenerators of S. We have $H_S = H_{S_1} H_{S_2}$, $\langle S \rangle = \langle S_1 \rangle \langle S_2 \rangle$, since the elements of S of degree d correspond to pairs (α, β), $|\alpha| = d_1$, $|\beta| = d_2$, $d = d_1 + d_2$.

Moreover $e_S = e_{S_1} e_{S_2}$, $\delta_S = \delta_{S_1} + \delta_{S_2}$, $c_S = c_{S_1} + c_{S_2}$ (just apply the definition of multiplicity and dimension).

A horizontal splitting is always possible, unless S is reduced to $\{0\}$: if $\alpha \neq 0$ is an element of the staircase, let $S' = \{\beta \in S \mid \alpha \leq \beta\}$, and $S_1 = S \setminus S'$; then S_1 is a staircase, and S' is a translate of a staircase S_2 by α, $S' = \alpha + S_2$. The cogenerators of S_1 are obtained by deleting from the cogenerators of S all the multiples of α and adding α. The cogenerators of S_2 are obtained as follows: let $\{\beta_i\}$ be a set of cogenerators of S; then a set of cogenerators of S_2 is given by $\beta_i : \alpha$, where the operation $:$ is defined as follows: $(b_1, \ldots, b_m) : (a_1, \ldots, a_m) = (c_1, \ldots, c_m)$, where $c_i = \max(b_i - a_i, 0)$. The operation corresponds to the operation $I : J$ between ideals.

In this case $H_S = H_{S_1} + H_{S'} = H_{S_1} + T^{|\alpha|} H_{S_2}$, by the definition of the Hilbert-Poincaré series; moreover $\langle S \rangle = \langle S_1 \rangle + T^{|\alpha|} \langle S_2 \rangle$, $d_S = \max(d_{S_1}, d_{S_2})$, $c_S = \min(c_{S_1}, c_{S_2})$, $e_S = e_{S_1} + e_{S_2}$ if $d_{S_1} = d_{S_2}$, $e_S = e_{S_1}$ if $d_{S_1} > d_{S_2}$, $e_S = e_{S_2}$ if $d_{S_1} < d_{S_2}$.

The element α is uniquely identified by the splitting, (it is the minimal element of S') and is called the *pivot* of the splitting.

In our applications, we assume that α is smaller than one of the cogenerators of S (in multiplicative notation, properly divides it), and different from 0; this is possible unless the staircase has no cogenerators or all cogenerators are of degree one. In that case, both S_1 and S_2 are strictly smaller than S. This implies that any chain of such splittings must terminate.

Dimension and Multiplicity

A computation of the same type is possible when we do not need the whole Hilbert-Poincaré series, but only the codimension (or the dimension), or the codimension and the multiplicity.

In the case of the codimension, it is sufficient to remark that in a vertical splitting the codimension is the sum of the codimensions of the pieces, in a horizontal splitting the codimension of the staircase is the minimum of the codimensions of the pieces. The algorithm is simplified remarking that when we have found one branch of the computation that gives codimension c, we can abandon all the other branches that give higher codimension, and this can often be checked at an early stage. Moreover, instead of the original staircase, we can take its radical: a staircase that has as generators (non minimal) the same elements in which every positive element of $\mathbf{N}$ is substituted by 1.

The multiplicity in general can be computed together with the codimension remarking that for vertical splittings the multiplicity is the product of the multiplicities, and for horizontal splittings the multiplicity is equal to the sum of the multiplicities if the codimension of the two parts is the same, and it is the multiplicity of the part of lower codimension if the two codimensions are different.

The preliminary computation of the codimension (often simpler, since one can take the radical at once) allows to abandon earlier the branches with higher codimension. This has never been tested, but will be implemented in a short time.

4 Terminating the Algorithm

We can proceed in splitting the staircase until each piece is cogenerated by one element of degree 1, or by no element, but this is impractical. We terminate the splitting when we are reduced to a set of cogenerators consisting of some pure powers and a few elements, pairwise coprime, that are not pure powers.

The following theorem holds:

THEOREM 1. *Assume that a staircase S has a minimal set of cogenerators $\{\pi_1, \ldots, \pi_r, \mu_1, \ldots, \mu_s\}$, such that the π_i are pure powers, and the μ_j are mixed and pairwise coprime. Consider the partition of the variables*

$\{\Pi_0, \ldots, \Pi_s\}$, *where* Π_0 *is the set of the variables not appearing in the* μ_j, *and* Π_l *is the set of the variables appearing in* μ_l.
Then S *is split vertically according to* $\{\Pi_0, \ldots, \Pi_s\}$ *into* $s+1$ *staircases* $S_0, \ldots, S_s$, *and every* $[S_i]$ *contains at most one mixed power.*

The proof is immediate. Of course, the degenerate case that Π_0 is empty is possible.

The computation of the Hilbert-Poincaré series of these simple staircases is shown in the two following theorem:

THEOREM 2. *Let* S *be a staircase in* $\mathbf{N}^n$ *such that* $[S] = \{\pi_1, \ldots, \pi_m\}$, $\pi_i = x_i^{c_i}$.
Then $H_S = \prod(1 - T^{c_i})/(1 - T)^n$

The proof is obtained through a further vertical splitting in staircases in $\mathbf{N}$, that have either no cogenerators (they coincide with $\mathbf{N}$) or one cogenerator π_i (they coincide with $\{0, 1, \ldots, c_i - 1\}$).

In the first case $H_S = 1 + T + T^2 + \ldots + T^i + \ldots = (1 - T)^{-1}$; in the second case $H_S = 1 + T + \ldots + T^{c_i - 1} = (1 - T)^{-1}(1 - T^{c_i})$.

THEOREM 3. *Let* S *be a staircase in* $\mathbf{N}^{m+r}$, *and assume* $[S] = \{\pi_1, \ldots, \pi_m, \tau\}$, $\pi_i = x_i^{a_i}$, $\tau = x_1^{b_1} \cdots x_m^{b_m} y_1^{c_1} \cdots y_r^{c_r}$, $a_i > b_i > 0$, $c_j > 0$.
Then

$$H_S = \left(\prod(1 - T^{a_i}) - T^{|c|} \prod(T^{b_i} - T^{a_i})\right) / (1 - T)^{m+r}$$

where $|c| = \sum c_j$.

The proof is done by considering two special subcases.

If $r = 0$ consider the staircase S' cogenerated by $x_1^{a_1}, \ldots, x_m^{a_m}$ and split it with pivot $\alpha = x_1^{b_1} \cdots x_m^{b_m}$; then S' is disjoint union of S and $\alpha + (S' : \alpha)$, and $S' : \alpha$ is cogenerated by $x_1^{a_1 - b_1}, \ldots, x_n^{a_n - b_n}$; hence H_S is computed by difference.

If $m = 0$ then consider the staircase $S' = \mathbf{N}^r$ and split it with pivot $\beta = y_1^{c_1} \cdots y_r^{c_r}$. Then S' is disjoint union of S and $\beta + S'$, and H_S is computed by difference.

In the general case, split S with pivot $y_1^{c_1} \cdots y_r^{c_r}$; then $S = S_1 \cup (\gamma + S_2)$, $\gamma = y_1^{c_1} \cdots y_r^{c_r}$, S_1 cogenerated by $x_1^{a_1}, \ldots, x_m^{a_m}, y_1^{c_1} \cdots y_r^{c_r}$, S_2 cogenerated by $x_1^{a_1}, \ldots, x_m^{a_m}, x_1^{b_1} \cdots x_m^{b_m}$ and we are reduced to the two previous subcases.

Dimension and Multiplicity

If one is interested only in the dimension or in the multiplicity, these can be computed directly.

In the case of Theorem 2, the dimension is $n - m$ and the multiplicity is $\prod c_i$.

In the case of Theorem 3, we have two subcases:

— if $m = 0$ then the dimension is $r - 1$ and the multiplicity is $|c|$;

— if $m > 0$ then the dimension is r and the multiplicity is $\prod a_i - \prod(a_i - b_i)$.

The proof is immediate by inspection of the formulae given in theorems 2 and 3.

5 The Choice of a Splitting

The vertical splittings appear occasionally, but when they are possible in an early stage of the algorithm their importance is dramatic. They are not so easy to discover, so it is wise to search for them when we have a hint that one might exist. We will see an example in which the algorithm is exponential without vertical splittings, but becomes very simple if we look for them. With this example the other known algorithms perform badly.

5.1 Vertical Splittings

The search for the most general vertical splitting is easy, but often useless; probably it is useful only when the sum of the degrees of the non-pure powers is not much larger than the number of variables.

Here is an algorithm for finding the vertical splittings, S being the set of cogenerators of the staircase.

```
P=empty
REPEAT for T in S
  REPEAT for U in P
    IF GCD(T,U) /= 1 THEN
       T=lcm(T,U)
       delete U from P
  ADD T to P
```

At the end of the algorithm, P is a set of coprime power products; if P contains only one element, then no vertical splitting is possible; otherwise P describes the partition of the variables defining the splitting.

In practice, if in the course of the algorithm T contains all the variables, then we can directly exit the algorithm declaring that no vertical splitting is possible.

Since the set P has never more elements than the set of variables, the complexity cannot exceed $O(n^2m)$, where n is the number of variables and m the number of cogenerators, since the cost of a GCD or a lcm is at most n arithmetic comparisons.

The only type of vertical splitting that is always useful to look for, is when a variable does not appear in the mixed power products. This is recognizable if the lcm of the radical of all the mixed power products does not contain one of the variables, and this only costs nm. These splittings appear frequently, and considerably simplify the algorithm: they do not have influence on the combinatorial part, but the product of a sum of univariate polynomials is simpler to compute than the sum of the products, and the effect of an early splitting is precisely to allow to perform the former instead of the latter.

5.2 Horizontal Splittings: Choice of the Pivot

The horizontal splittings can always be found; an optimal strategy would be to find every time a splitting such that the two pieces have sets of cogenerators that have one half of the cogenerators of the original staircase. This is possible (and easy, see below) when there are two variables, but is impossible in general. A strategy that allows splittings as balanced as possible is very useful. The reason for looking for such a strategy is the following: the algorithm for one splitting is of quadratic complexity (the interreduction of $S : \alpha$); splitting the cost in two at every step is as good as possible.

Consider the case of two variables; then we have a set of cogenerators $\tau_i = x^{n_i}y^{m_i}$, $i = 1, \ldots, d$, and we assume that the n_i are in ascending order (and hence the m_i are in descending order). Then if we take $x^{n_{d/2}}$ as pivot, we split the staircase into two staircases with half as many cogenerators.

There are several possible heuristics for the choice of the pivot. The choice that has appeared to be the best is the following: choose a variable that appears in at least two mixed power products, and some power products in which the variable appears. Then choose as pivot the GCD of these power

products. In particular, choosing a random variable among those that appears in most mixed power products, and three random power products (or two if only two exist) among those that contain this variable, the practical performance is often quite satisfying. (In some special cases it seems useful not to choose the three power products at random, but to choose them in a way to have a larger GCD; this heuristics has however not yet been implemented.)

If no variable appears in more than one mixed power product, these are all coprime, and we can terminate the algorithm as described in the previous section.

Probably an uniform strategy like this one is not good in every case, or at every point of the algorithm, and the issue of good heuristics is widely open. However the results even with this rough strategy are quite good.

6 Interreducing a Staircase

The interreduction of a non-minimal set of cogenerators $A = \{\alpha_1, \ldots, \alpha_m\}$ of a staircase S is the costliest part of each step of the algorithm; considerably improved performances can hence be obtained by optimizing this step.

The simplest interreduction algorithm is a double loop:

```
FOR T1 in A DO
  FOR T2 in A-{T1} DO
    IF T1 divides T2 THEN delete T2 from A
```

The cost of this algorithm is m^2 divisibility tests (m being the number of A).

The double loop can be cut in two, if we preliminarly sort A in a way that T_1 can divide T_2 only if T_1 precedes T_2; hence, if for example A is sorted in increasing degree, one half of the comparisons can be avoided. The preliminary sorting is not costly anyway, but we can even arrange the algorithm in such a way that the staircases that are generated are already correctly sorted. In this case the cost is $m^2/2$ divisibility tests.

In the more general situation, this algorithm is probably the best one, but often better algorithms can be obtained using some existing information on the staircase.

If A is obtained from $A' = [S]$ adding one cogenerator T, then one can simply delete from A' all the multiples of T and adding T to the result; one can insert the new element in the correct order, and check only the elements that follow it for divisibility. In this case, the cost is m divisibility tests.

We discuss now the case where a staircase S is obtained from another staircase S' dividing by a power product α. We discuss separately three cases: α is a variable, α is a pure power, α is generic. The more special cases allow more efficient algorithms.

Assume that we have $[S] = \{\alpha_1, \ldots, \alpha_m\}$, (this implying that no α_i divides another α_j) and that the α are already correctly sorted.

In the first case, $\alpha = x_i$, we split A in two, $B = \beta_1, \ldots, \beta_r$, $C = \gamma_1, \ldots, \gamma_s$, the β_j not divisible by x_i, the γ_j divisible by x_i; define $\gamma'_j = \gamma_j/x_i$; then the β_j do not divide any of the γ'_l (they do not divide the $\gamma_l = x_i\gamma_l$), and do not divide each other; the γ'_l may divide some of the β_j but do not divide each other. Moreover, the γ'_l are in correct degree order. Hence it is sufficient to delete from the β_j those elements that are multiple of some γ'_l, and merge the resulting β_j with the γ'_l. If B, C have m_1, m_2 elements respectively, ($m = m_1 + m_2$) the cost is at most $m_1 m_2$ divisibility tests (and the cost of merging is at most m).

In the second case, $\alpha = x_i^r$, divide $\alpha_1, \ldots, \alpha_m$ into $r+1$ subsets $A_0, \ldots, A_r$ according to the divisibility by x_i^j (elements of A_j being divisible by x_i^j).

Divide the elements of A_j by x_i^j, obtaining A'_j. Then elements of A'_j are not divisible by elements of A'_k if $k \leq j$. Hence one can obtain the result as follows:

```
FOR i= r down to 1
  delete from A'(i-1) the multiples of elements of A'(i)
  merge in  A'(i-1) the elements of A'(i)
```

and at the end $A'(0)$ is a minimal set of cogenerators.

Let a_i be the cardinality od A_i; then the cost of the algorithm is $\sum_{i=0}^{r-1} a_i \sum_{j=i+1}^{r} a_j$, that is smaller than $a_0(\sum_{j=1}^{s} a_i)^2/2$.

In the third general case, we split $\alpha_1, \ldots, \alpha_m$ in two, $B = \beta_1, \ldots, \beta_r$, $C = \gamma_1, \ldots, \gamma_s$, the β_j coprime with α, the γ_j not coprime with α; define $C' = \{\gamma'_j = \gamma_j : \alpha\}$; then the elements of B do not divide any of the elements of C'. Sort and interreduce C' with the general algorithm, delete from B the elements mutiple of some element of C', then merge B and C': this is the result. If B, C have m_1, m_2 elements respectively, the cost is $m_1 m_2 + m_2^2/2$.

We have never accounted for the cost of merging, or of splitting, that is linear. The overall cost of the algorithm, even in the third case, is very small when the splitting is uneven (many elements coprime with the pivot), and this happens quite frequently in the hardest computations; the fact that the pivots are bad is indeed the reason why the overall algorithm is hard. Hence the improvement is essential for these hard examples. This happens when the cogenerators are easily coprime, for example when the degree of the cogenerators are low compared with the number of variables. This is precisely the case in which the computation of the Hilbert-Poincaré series is especially difficult with all the known algorithms.

7 Comparison with the Other Known Algorithms

The algorithms of [KP] and [Ho] coincide with the present algorithm when the pivot is chosen to be a variable.

The algorithm of [BCR] coincides with the present algorithm, in the following variant: choose one variable x_i, and take as pivot x_i^n, where n is the minimum degree in which x_i appears. The algorithm requires the computation of several splittings for choosing the best variable; the overhead

can be frequently reduced by special considerations. When there are several variables and none is especially good, and we have to consider them all, it seems that the cost of computing several splittings is difficult to recover by discovering a relatively better variable.

The algorithm of [BS] is related to ours, with a difference. A staircase S is represented as the difference set $S_1 \setminus S_2$, S_1 obtained removing one of the cogenerators of S, and the $S_2 = S_1 \setminus S$, the T-staircase contained in S_1 composed of the multiples of this cogenerator. Both staircases are simpler (in a different sense than ours) and the termination can be done as in our algorithm. The choice of the cogenerator to remove is guided by heuristics dependent on the ordering.

The "Quick and dirty dimension algorithm" of [BS] coincides with our dimension algorithm if the choice of the pivot is made choosing the first variable of the smallest power product (in a suitable ordering).

These remarks show that our algorithm is "potentially better" than the algorithms of [KP], [Ho] and [BCR] that are a particular case (and indeed the experiments confirm this remark).

For the comparison with the algorithm of [BS], we remark that with our interpretation their dimension algorithm is just a special case of our Hilbert-Poincaré series algorithm. For many examples we remark that their algorithm produces badly balanced splittings, hence it is expected to be worse. The experiments confirm this feeling. The algorithm seems to be superior only when there are few cogenerators of high degree in several variables.

The algorithm of [MM] has mainly theoretical interest and it is known to be practically inefficient.

8 Practical Performance

The algorithm as explained above was implemented in COMMON-LISP and included in AlPi, and in Pascal and included in CoCoA.

The practical comparison of algorithms implemented in a heterogeneous way is hard, since it is difficult to separate the effect of the algorithm and the effect of the clever implementation; moreover some tricks can considerably speed the algorithms, and sometimes a trick can be applied to an algorithm and not to another.

To allow a fair evaluation of the algorithm, in the COMMON-LISP implementation we have included an approximate measure of the complexity. We consider as unit of complexity an operation on power products, such as GCD or lcm.

The interreduction of $[S] \cup \{\tau\}$ requires m operations, if S has m cogenerators (we have to test which power products are multiple of τ and delete them). The interreduction of the staircase $S : \tau$ costs m^2 operations, as we have seen in section 6. The interreduction of pure powers is simpler, and is made separately, so we can take m being the number of mixed cogenerators of S. Hence a rough estimate of the complexity of a single step is m^2.

In our implementation we have put a counter that accumulates m^2, and at the end of the algorithm we can know the number of steps, the maximum recursion depth and the sum of m^2. These data are reported in the tables of the examples.

The algorithm of [BCR] is implemented in CoCoA; an experimental implementation of the present algorithm in CoCoA allows some comparisons. Unfortunately, the limited number of variables allowed in CoCoA does not allow the comparison with the more difficult examples, and moreover the implementation of the algorithm of [BCR] in CoCoA is a highly optimized version, while the implementation of the present algorithm is only experimental. Anyway the behaviour is quite good, and the performance of the new algorithm is worse only in some highly stuctured cases.

A simple modification of the COMMON-LISP implementation could be used to perform the algorithm of [BCR], but the comparison would be unfair since the special structure of the pivots allows many improvements that are impossible in the general algorithm.

A very small modification of the implementation in COMMON-LISP (only a few lines of code) implements the algorithm of [BS]. The perfor-

mance of this rough implementation is not good, compared with the timings given in [BS], and it is not clear if this is due to an optimization of the implementation or to improvements to the algorithm; one can compare anyway the experimental complexity data (the number of steps, the sum of the m^2) and the algorithm of [BS] appears to be inferior (sometimes dramatically inferior) in all but some special cases with very few cogenerators.

The COMMON-LISP sources are available by anonymous FTP at the address `gauss.dm.unipi.it` (131.114.6.55), in the directory `pub/alpi-cocoa/hilbert`, together with documentation and some files of examples.

We report some data about the computation of some examples. These are some staircases associated to Gröbner bases, and some random examples. With $R_{n,m,d}$ we denote a set of m random power products of degree d in n variables (a random power product being a random product of powers of random degree of a random variable, everything with uniform distribution).

The non-random examples are the Gröbner bases of the following ideals:

1) Gonnet example, [BGK], homogenized, with DegRevLex ordering;

2) Valla example (see [TD]), DegRevLex ordering;

3) [BS], n.4.1

4) [BS], n.4.2

5) [BGK], Butcher, homogenized, DegRevLex

6) [TD], Cohn-2 homogenized, DegRevLex

7) [BGK], Hairer 2, homogenized, DegRevLex, up to degree 10

8) [BGK], Hairer 2, homogenized, DegRevLex, up to degree 11

9) [BGK], Hairer 2, homogenized, DegRevLex, the first 904 elements found (as you understand, we did not find the whole Gröbner basis).

10) to 12) are $R_{50,477,2}$, $R_{100,20,10}$, $R_{10,40,20}$.

The data reported are for our algorithm (BCRT) and for our implementation of [BS] algorithm, and we report the computing time on a SUN Sparc-2, the depth of the iterations, the total number of the iterations, and the sum of m^2 as described above. We report the data for one run of the probabilistic algorithm; the experience is that the deviation is not large.

Our implementation of the [BS] algorithm does not work in AKCL with more than 500 monomials because of bind stack overflow. In this case we have used an implementation of a variant, that is usually better (instead of proceeding by induction on all the cogenerators of the staircase only the mixed cogenerators are considered; moreover induction instead of recursion is used). For this variant, the recursion depth no longer makes sense; we have

preceded the timings by a *. The example 10) was too long anyway, and we have interrupted it.

	#	BCRT	depth	iter.	$\sum m^2$	BS	depth	iter.	$\sum m^2$
1	854	23″	17	853	2636718	*4′22″		2371	208144503
2	88	1″31	11	131	30630	9″	88	599	224349
3	1372	1′47″	22	2919	8396358	*13′25″		5274	860272285
4	4785	44′32″	34	100241	208685423	*6h25′		829623	41053923691
5	157	2″15	12	197	64967	17″57	157	341	1307015
6	71	1″	13	121	19220	3″15	71	155	122701
7	437	16″	14	733	517435	2′30″	433	1433	27175765
8	567	24″	14	983	827951	*1′25″		1057	898150
9	904	1′13″	16	1653	2603711	*4′		1279	243426014
10	477	1′15″	41	4701	2345490	> 24h		$> 10^7$	
11	20	2″	17	397	1946	2′	14	16383	267082
12	40	15″	18	2535	78129	26″	40	5025	154333

For the examples 3) and 4) the timings given in [BS] for the new implementation on a SUN Sparc-1 are 3′06″ and 2h53′23″ respectively, and > 4 days and 14h32′24″ for the Macaulay version.

9 Two Elementary Examples

Here we show two simple examples (one from [BCR], the other from [BS]) and our computation is in both cases simpler. We take as pivot the GCD of three power products containing one most frequent variable.

Let $S = [x^4, x^3y^3, x^3y^2z, x^3yz^2, y^2t, yt^5]$. The possible sequences of pivots are either a) (y, y, x^3) or b) (y, x^3), or c) (y^2, y, x^3), with probability $(9/20, 9/20, 1/10)$. The computations run as follows:

a):

$\langle S\rangle = \langle x^4, y\rangle + T\langle x^4, x^3y^2, x^3yz, x^3z^2, yt, t^5\rangle =$
$\langle x^4, y\rangle + T(x^4, y\rangle + T^2\langle x^4, x^3y, x^3z, t\rangle =$
$\langle x^4, y\rangle + T(x^4, y\rangle + T^2\langle x^3, t\rangle + T^5\langle x, y, z, t\rangle.$

b):

$\langle S\rangle = \langle x^4, y\rangle + T\langle x^4, x^3y^2, x^3yz, x^3z^2, yt, t^5\rangle =$
$\langle x^4, y\rangle + T\langle x^3, yt, t^5\rangle + T^4\langle x, y^2, yz, z^2, t\rangle.$

c):

$\langle S\rangle = \langle x^4, y^2, x^3yz^2, yt^5\rangle + T^2\langle x^4, x^3y, x^3z, t\rangle =$
$\langle x^4, y\rangle + T\langle x^4, y, x^3z^2t^5\rangle + T^2\langle x^3, t\rangle + T^5\langle x^4, y, z, t\rangle.$

The [BCR] algorithm corresponds to the pivot sequence (x^3, y, y, y):

$\langle S\rangle = \langle x^3, y^2t, yt^5\rangle + T^3\langle x, y^3, y^2z, yz^2, y^2t, yt^5\rangle =$
$\langle x^3, y\rangle + T\langle x^3, yt, t^5\rangle + T^3\langle x, y\rangle + T^4\langle x, y^2, yz, z^2, yt, t^5\rangle =$
$\langle x^3, y\rangle + T\langle x^3, yt, t^5\rangle + T^3\langle x, y\rangle + T^4\langle x, y, z^2, t^5\rangle + T^5\langle x, y, z, t\rangle.$

The [BS] algorithm computes:

$\langle S\rangle = \langle x^3y^3, x^3y^2z, x^3yz^2, y^2t, yt^5\rangle - T^4\langle y^3, y^2z, yz^2, y^2t, yt^5\rangle =$
$\langle x^3y^2z, x^3yz^2, y^2t, yt^5\rangle - T^6\langle z, t\rangle - T^4\langle y^2z, yz^2, y^2t, yt^5\rangle + T^7\langle z, t\rangle =$
$\langle x^3yz^2, y^2t, yt^5\rangle - T^6\langle z, t\rangle - T^6\langle z, t\rangle - T^4\langle yz^2, y^2t, yt^5\rangle + T^7\langle z, t\rangle + T^7\langle z, t\rangle =$
$\langle y^2t, yt^5\rangle - T^6\langle yt, t^5\rangle - 2T^6\langle z, t\rangle - T^4\langle yz^2, y^2t, yt^5\rangle + T^7\langle yt, t^5\rangle + 2T^7\langle z, t\rangle.$

Here is the second example. Let $S = [ac, ab^2, a^2b, a^3, b^3d]$. Then the pivot is either a or b. In the first case,

$\langle S\rangle = \langle a, b^3d\rangle + T\langle c, b^2, ab, a^2\rangle,$

in the second case,

$\langle S\rangle = \langle b, ac, a^3\rangle + \langle c, b^2, ab, a^2\rangle.$

The [BCR] algorithm has as first pivot either a or c (both variables are excellent, in the terminology of [BCR]), and in the second case:

$\langle S\rangle = \langle c, ab^2, a^2b, a^3, b^3d\rangle + T\langle a, b^3d\rangle =$
$\langle a, c, b^3d\rangle + T\langle c, b^2, ab, a^3\rangle + T\langle a, b^3d\rangle.$

The algorithm of [BS] computes

$$\langle S\rangle = \langle ac, ab^2, a^2b, a^3\rangle - T^4\langle a\rangle =$$
$$\langle ac, ab^2, a^2b\rangle - T^3\langle b\rangle - T^4\langle a\rangle =$$
$$\langle ac, ab^2\rangle - T^3\langle c, b\rangle - T^3\langle b\rangle - T^4\langle a\rangle =$$
$$\langle ac\rangle - T^3\langle c\rangle - T^3\langle c, b\rangle - T^3\langle b\rangle - T^4\langle a\rangle$$

10 A Simple Bad Example

The computation of the Hilbert-Poincaré series in general, and even computing the dimension, is a problem that is harder than an NP-complete problem (see [BS]), hence bad examples are unavoidable.

Here we study a very simple example that has a very bad behaviour, unless we allow vertical splittings and randomized algorithms: avoiding the general vertical splittings, or taking a "natural" ordering of the variables requires exponential time. The example is the following:

$$I = (x_0x_1, x_1x_2, \ldots, x_{n-1}x_n)$$

A randomized algorithm splits the staircase horizontally in two staircases, one with $n-2$ and one with $n-3$ elements; these can be split vertically, and the expected lengths are in ratio $3:1$. Hence the expected complexity is polynomial.

If no vertical splittings are allowed, more than $2^{n/3}$ steps are necessary. Moreover, if we always choose as pivot the lowest (or highest) possible variable appearing in more than one monomial, the splittings are always bad, and no vertical splittings are useful.

The algorithm of [BS] in this case has the same type of behaviour; however their heuristics is in this case the worst possible, and even with vertical splittings the algorithm remains exponential.

Indeed, with 42 variables the example can be computed with our implementation in $2''$, (with 101 variables it takes $40''$); without vertical splittings in 42 variables it takes $6'$, and 13h47$'$ with the algorithm of [BS].

References

[BCR] Bigatti, A.M., Caboara, M., Robbiano, L.,*On the computation of Hilbert-Poincaré series*, AAECC Journal, vol. 2, 1991

[BS] Bayer-Stillman, *Computation of Hilbert functions*, J. of Symbolic Comp. vol. 14 (1992)

[BGK] Boege, W., Gebauer, R., Kredel, H. *Some examples for solving systems of algebraic equations by calculating Gröbner bases*, J. of Symbolic Comp. (1986)

[GN] Giovini, A., Niesi, G. *CoCoA: a user-friendly system for commutative algebra*, DISCO-90 Proceedings, LNCS 429, Springer Verlag (1990)

[Ho] Hollman, J.,*On the computation of the Hilbert series*, Latin 92, Saõ Paulo, LNCS 583, Springer Verlag (1992)

[KP] Kondrat'eva, M.V., Pankrat'ev, E.V.,*A recursive algorithm for the computation of Hilbert polynomial*, EUROCAL 87, LNCS 387, Springer Verlag (1987)

[MM] Möller, M., Mora, T.,*The Computation of the Hilbert Function*, EUROCAL 83, LNCS 162, Springer Verlag (1983)

[TD] Traverso, C., Donati, L.,*Experimenting the Gröbner basis algorithm with the AlPi system*, ISSAC 89 Proceedings, A. C. M. (1989)