A Slice Algorithm for corners and Hilbert-Poincaré series of monomial ideals

Bjarke Hammersholt Roune

ABSTRACT

We present an algorithm for computing the corners of a monomial ideal. The corners are a set of multidegrees that support the numerical information of a monomial ideal such as Betti numbers and Hilbert-Poincaré series. We show an experiment using corners to compute Hilbert-Poincaré series of monomial ideals with favorable results.

1. INTRODUCTION

We present an algorithm that computes the corners of a monomial ideal along with their Koszul simplicial complexes. This allows to compute Hilbert-Poincaré series, irreducible decomposition [5] and Koszul homology (as described e.g. in [7]).

In a sense the corners are (or includes) those places on a monomial ideal where something "interesting" happens, and the Koszul simplicial complex for a corner encodes the local information about precisely what is happening there. In asking a computational (or otherwise) question about monomial ideals it is then a reasonable instinct to think about whether knowing the corners and their Koszul simplicial complexes would aid in answering that question. Corners are then a potentially valuable tool in constructing algorithms, and the theoretical and practical value of the tool depends on the theoretical and practical performance of algorithms for corners.

A recent and indeed first theoretical advance in this direction is a reverse search algorithm [2] for corners. As a reverse search algorithm it computes the corners of a monomial ideal in no more space up to a constant factor than that required by the input and output and in polynomial time.

Our contribution in this paper is an algorithm for corners that shows good practical performance. We demonstrate this by comparing it to the best algorithm for computing Hilbert-Poincaré series. We have not yet determined the theoretical time complexity of our algorithm, though this is an issue that deserves attention.

We call our algorithm a slice algorithm because it is inspired by and similar to the Slice Algorithm for maximal standard monomials of a monomial ideal [10]. Parts of the algorithm require modification to allow computation of corners, especially the proofs, though in particular the proof of termination is unchanged because it concerns the properties of monomial ideals and slices and not what is actually being computed. The main new idea that allows the Slice Algorithm to be applied to corners is that corners of full support have special properties that allow them to satisfy the equations that the Slice Algorithm is based on while corners in general do not.

We wish to thank Eduardo Saenz-de-Cabezon and Anna Maria Bigatti for helpful discussions on these topics.

2. BACKGROUND AND NOTATION

Let I be a monomial ideal in some polynomial ring with indeterminates $x_1, \ldots, x_n$. Let $\mathbb{x} \stackrel{\text{def}}{=} x_1 \cdots x_n$. We write a monomial $x_1^{v_1} \cdots x_n^{v_n}$ as x^v where v is the *exponent vector*. The *colon* of two monomials is $x^u : x^v \stackrel{\text{def}}{=} x^{\max(u-v,0)}$ and we will have frequent use of the function $\pi(m) \stackrel{\text{def}}{=} m : \mathbb{x}$.

We can only very briefly cover the needed concepts. We recommend [8] for a more in-depth introduction.

2.1 Monomial ideals

A monomial ideal is an ideal generated by monomials. Then a monomial ideal I has a unique minimal set of (monic) monomial generators $\min(I)$. The *least common multiple* of two monomials $\operatorname{lcm}(x^v, x^v) \stackrel{\text{def}}{=} x^{\max(u,v)}$ and the *greatest common denominator* is $\gcd(x^u, x^v) = x^{\min(u,v)}$. The colon of a monomial ideal by a monomial is $I : m \stackrel{\text{def}}{=} \langle a \,|\, am \in I \rangle = \langle a : m \,|\, a \in \min(I) \rangle$.

We plot a monomial ideal in a diagram by the exponent vectors of the monomials in the ideal, such as seen in Figure 1. The surface displayed in such a diagram is known as the *staircase surface*, and the monomials on it are those $m \in I$ such that $\pi(m) \notin I$.

Define the *lcm lattice* of a monomial ideal I by $\operatorname{lat}(I) \stackrel{\text{def}}{=} \{\operatorname{lcm}(M) \,|\, M \subseteq \min(I)\}$ with lcm as the join and gcd as the meet of the lattice. So e.g. $\operatorname{lat}(\langle x^2, xy \rangle) = \{1, x^2, xy, x^2y\}$.

The $\mathbb{N}^n$-graded Hilbert-Poincaré series of I is the possibly infinite sum of all monomials that are not in I. This sum can be written as a fraction with $(1 - x_1) \cdots (1 - x_n)$ in the denominator and a polynomial $H(I)$ in the numerator. When we talk of computing the Hilbert-Poincaré series of I in this paper we are talking about computing $H(I)$. There is also the more conventional total degree-graded Hilbert-Poincaré series, which is obtained by substituting $x_i \mapsto t$ for

each variable in the $\mathbb{N}^n$-graded Hilbert-Poincaré series.

A monomial ideal I is (weakly) *generic* [9] if whenever $x^u, x^v \in \min(I)$ and $u_i = v_i > 0$ for some i, then either $u = v$ or there is some third generator in $\min(I)$ that strictly divides $\operatorname{lcm}(x^u, x^v)$.

2.2 Simplicial complexes

An (abstract) *simplicial complex* Δ is a set of finite sets that is closed with respect to subset, i.e. if $v \in \Delta$ and $u \subseteq v$ then $u \in \Delta$. The elements of Δ are *faces*, and the inclusion-maximal faces are called *facets*. The set of facets is then

$$\operatorname{fac}(\Delta) \stackrel{\text{def}}{=} \{v \in \Delta \,|\forall u \in \Delta : v \subseteq u \Rightarrow v = u\}$$

The faces in this paper are all subsets of $\{x_1, \ldots, x_n\}$. The product of a face v is then $\Pi v \stackrel{\text{def}}{=} \Pi_{x_i \in v} x_i$ and the intersection of a set of faces V is $\cap V \stackrel{\text{def}}{=} \cap_{v \in V} v$.

The (upper) *Koszul simplicial complex* of a monomial ideal I at a monomial m is defined by

$$\Delta_m^I \stackrel{\text{def}}{=} \left\{ v \subseteq \{x_1, \ldots, x_n\} \,\middle|\, \frac{m}{\Pi v} \in I \right\},$$

where $\frac{m}{\Pi v} \notin I$ when Πv does not divide m. So for $I \stackrel{\text{def}}{=} \langle x^2, xy \rangle$ we see that $\Delta_{x^2y}^I = \{\emptyset, \{x\}, \{y\}\}$ and $\Delta_{x^2}^I = \{\emptyset\}$.

We remark that Δ_m^I encodes the shape of the stairface surface of I around m. This yields interesting information about I at m, e.g. Δ_m^I determines the Betti numbers at m.

A monomial m is a *corner* of a monomial ideal I when no variable lies in every facet of Δ_m^I. The set of corners is then

$$\operatorname{cor}(I) \stackrel{\text{def}}{=} \left\{ \text{monomials } m \,\middle|\, \cap \operatorname{fac}\left(\Delta_m^I\right) = \emptyset \right\}.$$

We do not consider m to be a corner if $\Delta_m^I = \emptyset$, while m is a corner if $\Delta_m^I = \{\emptyset\}$. So e.g. $\operatorname{cor}\left(\langle x^2, xy \rangle\right) = \left\{x^2, xy, x^2y\right\}$.

The corners can be identified from a diagram of a monomial ideal as those points where the staircase surface is bent in every axis direction. The reader may verify that the corners lie on both the lcm lattice and the staircase surface.

As pointed out in [2], all multidegrees that have homology are corners. So knowing the set of corners and their Koszul simplicial complexes allows to determine interesting information such as Betti numbers.

3. THE SLICE ALGORITHM IN BRIEF

The Slice Algorithm we present here is a divide and conquer algorithm that computes the corners of a monomial ideal along with their Koszul simplicial complexes.

As a divide and conquer algorithm, the Slice Algorithm breaks the problem it is solving into two problems that are more easily solved. This process continues recursively until the problems are *base cases*, i.e. they are easy enough that they can be solved directly. The minimal ingredients of the Slice algorithm are then a recursive step, a base case, a proof of termination and a proof of correctness.

4. THE RECURSIVE STEP

The Slice Algorithm operates on what we call *slices*. A slice A represents a subset of the corners of the input ideal, and we refer to this subset as the *content* $\operatorname{con}(A)$ of the slice. The Slice Algorithm recursively splits a slice A into two less complicated slices B and C such that $\operatorname{con}(A)$ is the disjoint union of $\operatorname{con}(B)$ and $\operatorname{con}(C)$.

We first present the formal definition of a slice and the equation we use to split slices. We follow that by an example that suggests a visual intuition of what the equation is stating. After that we prove that the equation is correct.

Definition 1. A *slice* is a 3-tuple (I, S, q) where I and S are monomial ideals and q is a monomial. The *content* of (I, S, q) is defined by

$$\operatorname{con}(I, S, q) \stackrel{\text{def}}{=} \left\{ \left(mq, \Delta_{m\mathbb{x}}^I\right) \,\middle|\, m\mathbb{x} \in \operatorname{cor}(I) \text{ and } m \notin S \right\}.$$

The Slice Algorithm computes content, and this suffices to compute corners since $\operatorname{cor}(I) = \operatorname{con}(I\mathbb{x}, \langle 0 \rangle, 1)$. Note how the multiplication by $\mathbb{x}$ in $I\mathbb{x}$ and in the definition of content cancel each other out. This might seem to be a superfluous complication that we could resolve by simply removing $\mathbb{x}$ in both places. However, the significance of $\mathbb{x}$ in the definition of content is that we consider only corners of full support. Corners of full support have special properties that the Slice Algorithm depends on. This can be seen by the fact that many of our lemmas impose a condition of full support and that those lemmas cease to hold if the condition is lifted.

If C is a set of pairs (m, Δ) and S is a monomial ideal, then it will be of considerable convenience for us to perform set operations between C and S while not paying attention to the simplicial complexes in C. I.e.

$$C \cap S \stackrel{\text{def}}{=} \{(m, \Delta) \in C \,|m \in S\},$$
$$C \setminus S \stackrel{\text{def}}{=} \{(m, \Delta) \in C \,|m \notin S\}.$$

The Slice Algorithm uses the following equation to split a slice into two less complicated slices. We illustrate this in Example 1 and we discuss it further after the example.

$$\operatorname{con}(I, S, q) = \operatorname{con}(I : p, S : p, qp) \;\cup\; \operatorname{con}(I, S + \langle p \rangle, q).$$

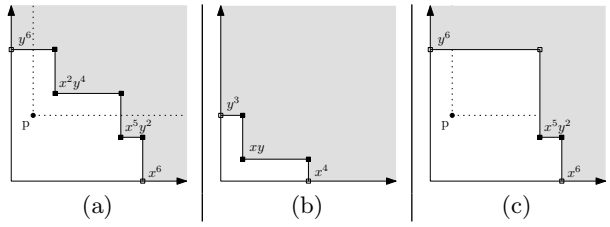


Figure 1: Illustrations for example 1.

Example 1. Let $I := \langle x^6, x^5y^2, x^2y^4, y^6 \rangle$ and $p := xy^3$. Then I is the ideal depicted in Figure 1(a) where $\langle p \rangle$ is indicated by the dotted line. The corners are indicated by squares, and the squares for the corners of full support are filled. The full support corners are

$$\left\{x^2y^6, x^2y^4, x^5y^4, x^5y^2, x^6y^2\right\}.$$

We compute this set of full support corners by performing a step of the Slice Algorithm. We will not mention the Koszul upper complexes, but the reader may verify that these work out correctly as well.

Let I_1 be the ideal $I : p = \langle y^3, xy, x^4 \rangle$, as depicted in Figure 1(b). As can be seen by comparing figures 1(a) and 1(b), the ideal I_1 corresponds to the part of the ideal I that lies within $\langle p \rangle$. Thus it is reasonable to expect that the full support corners of I_1 corresponds (after multiplication by

p) to the full support corners of I that lie within $\langle p\rangle$. This turns out to be true, since

$$\left\{xy^3, xy, x^4y\right\} * p = \left\{x^2y^6, x^2y^4, x^5y^4\right\}.$$

It now only remains to find the full support corners of I that lie outside of $\langle p\rangle$. Let $I_2 := \langle x^6, x^5y^2, y^6\rangle$ as depicted in Figure 1(c). The dotted line indicates that we are ignoring everything inside $\langle p\rangle$. It happens to be that one of the minimal generators of I, namely x^2y^4, lies in the interior of $\langle p\rangle$, which allows us to ignore that minimal generator. We see that the corners of full support of I_2 that lie outside of $\langle p\rangle$ are $\left\{x^5y^2, x^6y^2\right\}$.

We have now found all the full support corners of I from the full support corners of I_1 and those full support corners of I_2 that lie outside of $\langle p\rangle$. Using the language of slices, we have split the slice $A := (I, \langle 0\rangle, 1)$ into the two slices $A_1 := (I_1, \langle 0\rangle, p)$ and $A_2 := (I_2, \langle p\rangle, 1)$, and indeed

$$\begin{aligned}\operatorname{con}(A) &= \left\{x^2y^6, x^2y^4, x^5y^4, x^5y^2, x^6y^2\right\}\\ &= \left(\left\{xy^3, xy, x^4y\right\} * xy^3\right) \cup \left\{x^5y^2, x^6y^2\right\}\\ &= \operatorname{con}(A_1) \cup \operatorname{con}(A_2),\end{aligned}$$

where the union is disjoint.

What we did in Example 1 was to rewrite $\operatorname{con}(I, S, q)$ as

$$\operatorname{con}(I, S, q) = (\operatorname{con}(I, S, q) \cap \langle qp\rangle) \ \cup \ (\operatorname{con}(I, S, q) \setminus \langle qp\rangle),$$

and we wrote the two disjoint sets on the right hand side of this equation as the content of two slices. We now seek a way to do this given a general slice (I, S, q) and a polynomial p. This is easy to do for the second set on the right hand side, since the definition of content implies that

$$\operatorname{con}(I, S + \langle p\rangle, q) = \operatorname{con}(I, S, q) \setminus \langle qp\rangle.$$

For the first set on the right hand side, we refer to Theorem 1, which states that

$$\operatorname{con}(I : p, S : p, qp) = \operatorname{con}(I, S, q) \cap \langle qp\rangle.$$

Example 1 gives an intution of why this should be true.

Putting together the pieces, we get the *pivot split equation*

$$\operatorname{con}(I, S, q) = \operatorname{con}(I : p, S : p, qp) \ \cup \ \operatorname{con}(I, S + \langle p\rangle, q). \tag{1}$$

This equation is the basic engine of the Slice Algorithm. We will discuss it and its parts at length, so we introduce names to make such discussion convenient. The process of applying the pivot split equation is called a *pivot split* and p is the *pivot*. The left hand side slice (I, S, q) is the *current slice*, since it is the slice we are currently splitting. The first right hand slice $(I : p, S : p, qp)$ is the *inner slice*, since its content is inside $\langle qp\rangle$. The second right hand slice $(I, S + \langle p\rangle, q)$ is the *outer slice*, since its content is outside $\langle qp\rangle$.

We have stated that the Slice Algorithm splits a slice into two less complicated slices. So both the inner slice and the outer slice should be less complicated than the current slice. This is so for the inner slice because $I : p$ generally is a less complicated monomial ideal than I is. It is not immediately clear that the outer slice $(I, S + \langle p\rangle, q)$ is less complicated than the current slice. To see how it can be less complicated, consider Equation (2) which we prove in Theorem 1.

$$\operatorname{cor}(I) \setminus S = \operatorname{cor}\left(I'\right) \setminus S, \quad I' \stackrel{\text{def}}{=} \langle m \in \min(I) \,|\, \pi(m) \notin S\rangle. \tag{2}$$

This equation states that we can remove from $\min(I)$ those elements that are strictly divisible by some element of S without changing the content of the slice. The outer slice has $S + \langle p\rangle$ where the current slice has S, so there is the potential to remove elements of $\min(I)$ due to Equation (2).

We apply Equation (2) whenever it is of benefit to do so, which it is when $\pi(\min(I)) \cap S \neq \emptyset$. Otherwise we say that the slice is *normal*, i.e. when $\pi(\min(I)) \cap S = \emptyset$.

THEOREM 1. *If p is a monomial, then*

i) $\operatorname{con}(I : p, S : p, qp) = \operatorname{con}(I, S, q) \cap \langle qp\rangle$,

ii) $\operatorname{con}(I, S, q) = \operatorname{con}(I', S, q)$,

where $I' \stackrel{def}{=} \langle m \in \min(I) \,|\, \pi(m) \notin S\rangle$.

PROOF. ***i)*:** We get from the definition of content that

$$\begin{aligned}&\operatorname{con}(I : p, S : p, qp)\\ &\quad= \left\{\left(mpq, \Delta^{I:p}_{m\mathbb{x}}\right) | m\mathbb{x} \in \operatorname{cor}(I : p) \text{ and } m \notin S : p\right\}\\ &\quad= \left\{\left(m'q, \Delta^{I:p}_{m'\mathbb{x}:p}\right) \,\middle|\, \begin{array}{l} p \text{ divides } m' \text{ and}\\ m' : p \notin S : p \text{ and}\\ m'\mathbb{x} : p \in \operatorname{cor}(I : p)\end{array}\right\}\\ &\operatorname{con}(I, S, q) \cap \langle qp\rangle\\ &\quad= \left\{\left(mq, \Delta^{I}_{m\mathbb{x}}\right) | p \text{ divides } m \notin S \text{ and } m\mathbb{x} \in \operatorname{cor}(I)\right\}\end{aligned}$$

We prove that the two sets are equal by showing that each pair of similar conditions above are in fact equivalent. Even though m and m' are the same monomial, we retain the distinction to make it clear which set we are referring to. Going from left to right, we get by Lemma 1 that $\Delta^{I}_{m\mathbb{x}} = \Delta^{I:p}_{m'\mathbb{x}:p}$. This leaves only the rightmost condition about corners.

Whether $m\mathbb{x}$ is an element of $\operatorname{cor}(I)$ depends only on $\Delta^{I}_{m\mathbb{x}}$. Likewise, whether $m'\mathbb{x} : p$ is an element of $\operatorname{cor}(I : p)$ depends only on $\Delta^{I:p}_{m'\mathbb{x}:p}$. We have just seen that these two simplicial complexes are equal, so $m\mathbb{x}$ is an element of $\operatorname{cor}(I)$ if and only if $m'\mathbb{x} : p$ is an element of $\operatorname{cor}(I : p)$. This leaves only the matter of $m \notin S$ being equivalent to $m' : p \notin S : p$.

If t is a monomial such that $p|t$ then $t \in S \Leftrightarrow t : p \in S : p$, so $m \notin S$ if and only if $m' : p \notin S : p$ and we are done.

***ii)*:** Lemma 3 implies the more general statement that if $\pi(I) \setminus S = \pi(I') \setminus S$ then $\operatorname{con}(I, S, q) = \operatorname{con}(I', S, q)$. The former equation is satisfied by the particular I and I' in the theorem since it holds for monomials a that

$$\begin{aligned}&a \in \pi\left(I'\right) \setminus S \Leftrightarrow \exists m \in \min\left(I'\right) : \pi(m) \,|\, a \text{ and } a \notin S\\ &\quad\Leftrightarrow \exists m \in \min(I) : \pi(m) \notin S \text{ and } \pi(m) \,|\, a \text{ and } a \notin S\\ &\quad\Leftrightarrow \exists m \in \min(I) : \pi(m) \,|\, a \text{ and } a \notin S \Leftrightarrow a \in \pi(I) \setminus S. \quad \square\end{aligned}$$

LEMMA 1. *If $p|m$ then $\Delta^{I}_{m\mathbb{x}} = \Delta^{I:p}_{m\mathbb{x}:p}$.*

PROOF. We use Lemma 2 with $A \stackrel{\text{def}}{=} I$, $B \stackrel{\text{def}}{=} (I : p)p$ and $c \stackrel{\text{def}}{=} m\mathbb{x}$. The preconditions of Lemma 2 are satisfied since $\langle \pi(m\mathbb{x})\rangle = \langle m\rangle \subseteq \langle p\rangle$ and $A \cap \langle p\rangle = B \cap \langle p\rangle$, so

$$\Delta^{I}_{m\mathbb{x}} = \Delta^{(I:p)p}_{m\mathbb{x}} = \Delta^{(I:p)p}_{(m\mathbb{x}:p)p} = \Delta^{I:p}_{m\mathbb{x}:p}. \quad \square$$

LEMMA 2. *If A and B are monomial ideals and c is a monomial such that $A \cap \langle \pi(c)\rangle = B \cap \langle \pi(c)\rangle$, then $\Delta^{A}_{c} = \Delta^{B}_{c}$.*

PROOF. Let $v \in \Delta^{A}_{c}$. Then $\pi(c) \,|\, \frac{c}{\Pi v} \in A$ so $\frac{c}{\Pi v} \in A \cap \langle \pi(c)\rangle = B \cap \langle \pi(c)\rangle$ so $\frac{c}{\Pi v} \in B$ so $v \in \Delta^{B}_{c}$. Swap A and B in this proof to get the other inclusion. $\square$

LEMMA 3. *If A, B and C are monomial ideals such that $\pi(A) \setminus C = \pi(B) \setminus C$ and $m \notin C$ is a monomial, then $\Delta^A_{m\mathbb{x}} = \Delta^B_{m\mathbb{x}}$.*

PROOF. Let $v \in \Delta^A_{m\mathbb{x}}$. Then $\frac{m\mathbb{x}}{\Pi v} \in A$ so $\frac{m}{\Pi v} \in \pi(A)$. As $\frac{m}{\Pi v}|m \notin C$ this implies that $\frac{m}{\Pi v} \in \pi(A)\backslash C = \pi(B)\backslash C$. Then $\frac{m}{\Pi v} \in \pi(B)$ so $\frac{m\mathbb{x}}{\Pi v} \in \pi(B)\,\mathbb{x} = B \cap \langle \mathbb{x} \rangle \subseteq B$ so $v \in \Delta^B_{m\mathbb{x}}$. Swap A and B in this proof to get the other inclusion. □

5. THE BASE CASE

In this section we present the base case of the Slice Algorithm. A slice (I, S, q) is a *base case slice* if I is square free or if I does not have full support (i.e. $\mathbb{x}$ does not divide $\operatorname{lcm}(\min(I))$). Theorem 2 and Theorem 3 show how to obtain the content of a base case slice.

THEOREM 2. *If I is a monomial ideal that does not have full support, then* $\operatorname{con}(I, S, q) = \emptyset$.

PROOF. No element of the lcm lattice of I has full support when I does not have full support. The corners of I lie on the lcm lattice of I, and the only corners of I we consider for the content are those of full support. □

Recall that ϕ maps sets $v \subseteq \{x_1, \ldots, x_n\}$ to the product of variables not in v, i.e. $\phi(v) = \Pi\bar{v} = \Pi_{x_i \notin v} x_i$. The main fact to keep in mind about ϕ is that it maps a subset relation into a domination relation, i.e. $v \supseteq u \Leftrightarrow \phi(v)|\phi(u)$.

THEOREM 3. *If (I, S, q) is a slice such that I is square free and has full support, then* $\operatorname{con}(I, S, q) = \left\{(q, \Delta^I_{\mathbb{x}})\right\}$ *where* $\operatorname{fac}\left(\Delta^I_{\mathbb{x}}\right) = \phi^{-1}(\min(I))$.

PROOF. Lemma 4 implies that

$$\phi\left(\operatorname{fac}\left(\Delta^I_{\mathbb{x}}\right)\right) = \min(I\mathbb{x} : \mathbb{x}) \setminus \left\langle x_1^2, \ldots, x_n^2 \right\rangle = \min(I).$$

This implies that

$$\phi\left(\cap \operatorname{fac}\left(\Delta^I_{\mathbb{x}}\right)\right) = \operatorname{lcm}\left(\phi(\operatorname{fac}\left(\Delta^I_{\mathbb{x}}\right)\right) = \operatorname{lcm}(\min(I)) = \mathbb{x}.$$

Then $\cap \operatorname{fac}\left(\Delta^I_{\mathbb{x}}\right) = \phi^{-1}(\mathbb{x}) = \emptyset$ so $\mathbb{x}$ is a corner of I. The corners of I lie on the lcm lattice, so they are all square free. We only consider corners of full support for the content, so $\mathbb{x}$ is the only corner that appears in the content. □

LEMMA 4. *If m is a monomial, then*

$$\phi\left(\operatorname{fac}\left(\Delta^I_m\right)\right) = \min(I\mathbb{x} : m) \setminus \left\langle x_1^2, \ldots, x_n^2 \right\rangle.$$

PROOF. We see that $\phi(\operatorname{fac}\left(\Delta^I_m\right)) = \min(\phi(\Delta^I_m))$. Then the result follows by applying min to both sides of

$$\phi(\Delta^I_m) = \{a \in I\mathbb{x} : m \mid a \text{ is a square free monomial}\}.$$

Every square free monomial can be written as $\phi(v)$ for some $v \subseteq \{x_1, \ldots, x_n\}$, so this equation follows from

$$\begin{aligned}\phi(v) \in \phi(\Delta^I_m) &\Leftrightarrow v \in \Delta^I_m \Leftrightarrow \frac{m}{\Pi v} \in I \\ &\Leftrightarrow \frac{m\mathbb{x}}{\Pi v} \in I\mathbb{x} \Leftrightarrow m\phi(v) \in I\mathbb{x} \\ &\Leftrightarrow \phi(v) \in I\mathbb{x} : m. \quad \square\end{aligned}$$

6. TERMINATION

We present four conditions on the choice of the pivot in pivot splits that are necessary and jointly sufficient to ensure termination. Each condition is independent of the others.

The conditions are listed below, along with an explanation of why violating any one of the conditions results in an inner or outer slice that is equal to the current slice. Once that happens the split can be repeated forever so that the Slice Algorithm would not terminate, so this shows that each condition is necessary. Note that just the first two conditions are sufficient to ensure termination at this point, but the last two conditions will become necessary after some of the improvements in Section 9 are applied.

Condition 1: $p \notin S$
Otherwise $p \in S$ and then the outer slice will be equal to the current slice.

Condition 2: $p \neq 1$
Otherwise $p = 1$ and then the inner slice will be equal to the current slice.

Condition 3: $p \notin I$
Otherwise the outer slice will be equal to the current slice after "Pruning of S" from Section 9.

Condition 4: $p|\pi(\operatorname{lcm}(\min(I)))$
Otherwise the outer slice will be equal to the current slice after "More pruning of S" from Section 9.

We say that a pivot is *valid* when it satisfies these four conditions. Having imposed these conditions, we need to show that every slice that is not a base case admits a valid pivot (Theorem 4), and that it is not possible to keep splitting on valid pivots forever (Theorem 5).

THEOREM 4. *If (I, S, q) is normal and admits no valid pivot, then I is square free and so (I, S, q) is a base case.*

PROOF. Suppose I is not square free. Then there exists an x_i such that $x_i^2|m$ for some $m \in \min(I)$, which implies that $x_i \notin I$. Also, $x_i \notin S$ since $x_i|\pi(m)$ and (I, S, q) is normal. We conclude that x_i is a valid pivot. □

THEOREM 5. *Selecting valid pivots ensures termination.*

PROOF. The polynomial ring we are working within is noetherian, i.e. it does not contain an infinite sequence of ideals that is strictly increasing. We show that if the Slice Algorithm does not terminate, then such a sequence exists.

Let f and g be functions mapping slices to ideals, and define them by $f(I, S, q) \stackrel{\text{def}}{=} S$ and $g(I, S, q) \stackrel{\text{def}}{=} \langle \operatorname{lcm}(\min(I)) \rangle$.

Suppose we split a non-base case slice A where A_1 is the inner slice and A_2 is the outer slice. Then Condition 1, Condition 2 and the fact that I has full support imply that

$$\begin{aligned} f(A) &\subseteq f(A_1), & g(A) &\subsetneq g(A_1), \\ f(A) &\subsetneq f(A_2), & g(A) &\subseteq g(A_2). \end{aligned}$$

Also, if we let A be an arbitrary slice and we let A' be the corresponding normal slice, then

$$f(A) \subseteq f(A'), \qquad g(A) \subseteq g(A').$$

So we see that f and g never decrease, and one of them strictly increases on the outer slice while the other strictly increases on the inner slice. Thus there does not exist an infinite sequence of splits on valid pivots. □

7. PSEUDO CODE

We show the Slice Algorithm in pseudo code.

function $\operatorname{con}(I, S, q)$

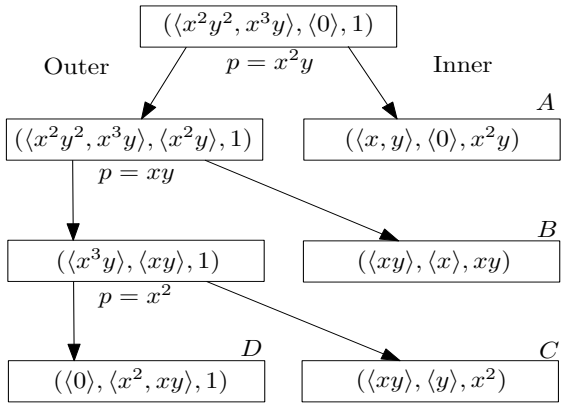
let $I' \stackrel{\text{def}}{=} \langle m \in \min(I) \,|\, \pi(m) \notin S \rangle$
if $\mathbb{x}$ does not divide $\operatorname{lcm}(\min(I'))$ **then return** $\emptyset$
if I' is square free **then return** $\{(q, \phi^{-1}(\min(I)))\}$
let p be some monomial such that $1 \neq p \notin S$
return $\operatorname{con}(I' : p, S : p, qp) \cup \operatorname{con}(I', S + \langle p \rangle, q)$

We have represented the simplicial complexes by their facets, so $\operatorname{con}(I\mathbb{x}, \langle 0 \rangle, 1)$ returns $\{(m, \operatorname{fac}(\Delta^I_m)) \,|\, m \in \operatorname{cor}(I)\}$.

For the choice of p an easy though inefficient choice is to follow the proof of Theorem 4 and select an x_i such that x_i^2 divides some minimal generator of I'.

8. EXAMPLE

The tree shows the steps of the algorithm on $\langle xy, x^2 \rangle$.



The contents of the leaves are (we specify facets only)

$$\operatorname{con}(A) = \{(x^2y, \{\{y\}, \{x\}\})\}, \quad \operatorname{con}(B) = \{(xy, \{\emptyset\})\},$$
$$\operatorname{con}(C) = \{(x^2, \{\emptyset\})\}, \quad \operatorname{con}(D) = \emptyset.$$

9. IMPROVEMENTS

In this section we show a number of improvements to the basic version of the Slice Algorithm presented so far.

It is natural that more specific versions of the improvements presented here also apply to the Slice Algorithm for maximal standard monomials and irreducible decomposition [10]. We use this fact in reverse by transferring the improvements to that algorithm to our current setting of corners and Koszul simplicial complexes. The improvements that rely only on the properties of monomial ideals and slices apply without change, while those that rely in their essence on the particular definition of content have to be adapted.

We summarize and classify each improvement according to whether it needs to be adapted. We refer to [10] for more detail on those improvements that apply without change.

Monomial lower bounds on slice contents: It is possible to replace a slice by a simpler slice with the same content using a monomial lower bound on the content. This improvement relies on the definition of content and so has to be adapted to apply to our setting.

Independence splits: This improvement applies to monomial ideals that have independent sets of variables. We briefly describe how to adapt this to our setting.

A base case for two variables: There is a base case for ideals in two variables. This improvement has to be adapted to our setting.

Pruning of S: If (I, S, q) is a slice, this improvement is to remove elements of $\min(S)$ that lie in I. This can speed things up in case $|\min(S)|$ becomes large. The improvement and its proof apply without change.

More pruning of S: If (I, S, q) is a slice, this improvement is to remove elements of $\min(S)$ that do not strictly divide $\operatorname{lcm}(\min(I))$. A significant implication of this is that pivots that are pure powers can always be removed from S after normalization. The improvement and its proof apply without change.

Minimizing the inner slice: This is a general monomial ideal technique for fast calculation of colons and intersections of a general monomial ideal by a principal monomial ideal. This applies to computing inner slices. The technique applies without change.

Reduce the size of exponents: This is a general monomial ideal technique for supporting arbitrary precision exponents in a way that is as fast as using native machine integers. The technique applies without change.

9.1 Monomial lower bounds on slice contents

Let l be a *monomial lower bound* on the slice (I, S, q) in the sense that $ql|c$ for all $c \in \operatorname{con}(I, S, q)$. In a pivot split on l, we can then predict that the outer slice will be empty. So the Pivot Split Equation (1) specializes to

$$\operatorname{con}(I, S, q) = \operatorname{con}(I : l, S : l, ql), \tag{3}$$

which shows that we can get the effect of performing a split while only having to compute a single slice. This is only interesting if we can determine a lower bound of a slice without already knowing its content, which is what Theorem 6 does.

THEOREM 6. *If (I, S, q) is a slice, then*

$$l_{x_i} \stackrel{def}{=} \pi(\gcd(\min(I) \cap \langle x_i \rangle))$$

is a monomial lower bound on (I, S, q) for each variable x_i.

PROOF. Suppose $c \in \operatorname{cor}(I)$ such that $x_i|c$. As c lies on the lcm lattice there is then an $m \in \min(I)$ such that $x_i|m|c$ and then $\gcd(\min(I) \cap \langle x_i \rangle)|m|c$.

If $qc \in \operatorname{con}(I, S, q)$ then $c\mathbb{x} \in \operatorname{cor}(I)$, and we have just proven that this implies that $l_{x_i} = \pi(\cdots) \,|\, \pi(c\mathbb{x}) = c$. □

Theorem 6 allows us to make a slice simpler with no change to the content, and this can be iterated until a fixed point is reached simultaneously for every variable.

9.2 Independence splits

Following [10] we define I-independence and show how this allows to perform a more efficient kind of split. This technique for the Slice Algorithm was inspired by a similar technique for computing Hilbert-Poincaré series that was first suggested in [1] and described in more detail in [4].

Definition 2. Let A, B be non-empty disjoint sets such that $A \cup B = \{x_1, \ldots, x_n\}$. Then A and B are *I-independent* if $\min(I) \cap \langle A \rangle \cap \langle B \rangle = \emptyset$.

In other words, A and B are I-independent if no element of $\min(I)$ is divisible by both a variable in A and a variable in B. For this section, let R be the ambient polynomial ring of I, and let R_A be the subring with the variables from A and let R_B be the subring with the variables from B. These subrings then have their own versions ϕ_A and ϕ_B of ϕ, i.e.

$$\phi(v) \stackrel{\text{def}}{=} \frac{\mathbb{x}}{\Pi v}, \qquad \phi_A(v) \stackrel{\text{def}}{=} \frac{\Pi A}{\Pi v}, \qquad \phi_B(v) \stackrel{\text{def}}{=} \frac{\Pi B}{\Pi v},$$

and we project I to $I_A \stackrel{\text{def}}{=} I \cap R_A$ and $I_B \stackrel{\text{def}}{=} I \cap R_B$.

Example 2. Let $I \stackrel{\text{def}}{=} \langle x^2, xy, y^2, z^2 \rangle$, $A \stackrel{\text{def}}{=} \{x, y\}$ and $B \stackrel{\text{def}}{=} \{z\}$. Then A and B are I-independent. We have that

$$\begin{aligned}
\operatorname{fac}\left(\Delta^I_{xyz}\right) &= \{\{z\}\} && \mapsto^{\phi} \{xy\}, \\
\operatorname{fac}\left(\Delta^{I_A}_{xy}\right) &= \{\emptyset\} && \mapsto^{\phi_A} \{xy\}, \\
\operatorname{fac}\left(\Delta^{I_B}_{z}\right) &= \{\} && \mapsto^{\phi_B} \{\},
\end{aligned}$$

and so we get Δ^I_{xyz} in terms of its projections as

$$\phi\left(\operatorname{fac}\left(\Delta^I_{xyz}\right)\right) = \phi_A\left(\operatorname{fac}\left(\Delta^{I_A}_{xy}\right)\right) \cup \phi_B\left(\operatorname{fac}\left(\Delta^{I_B}_{z}\right)\right).$$

We get the corresponding equation at xyz^2 since

$$\begin{aligned}
\operatorname{fac}\left(\Delta^I_{xyz^2}\right) &= \{\{x, y\}, \{z\}\} && \mapsto^{\phi} \{z, xy\}, \\
\operatorname{fac}\left(\Delta^{I_B}_{z^2}\right) &= \{\emptyset\} && \mapsto^{\phi_B} \{z\}.
\end{aligned}$$

Theorem 7 generalizes the observation in Example 2. The process of applying Theorem 7 is called an *independence split*. The statement may seem more familiar if one considers that $\phi(\Delta)$ is the Alexander dual of the Stanley-Reisner ideal of Δ, though we offer a direct proof for completeness.[1]

THEOREM 7. *If A, B are I-independent and $a \in R_A$ and $b \in R_B$ are monomials such that ab has full support, then*

$$\phi\left(\operatorname{fac}\left(\Delta^I_{ab}\right)\right) = \phi_A\left(\operatorname{fac}\left(\Delta^{I_A}_{a}\right)\right) \cup \phi_B\left(\operatorname{fac}\left(\Delta^{I_B}_{b}\right)\right).$$

In particular,

$$\operatorname{cor}(I) \cap \langle \mathbb{x} \rangle = (\operatorname{cor}(I_A) \cap \langle \mathbb{x} \rangle) \cdot (\operatorname{cor}(I_B) \cap \langle \mathbb{x} \rangle).$$

PROOF. Let $a' \stackrel{\text{def}}{=} \pi(a)$ and $b' \stackrel{\text{def}}{=} \pi(b)$. Then $ab = a'b'\mathbb{x}$ and we apply Theorem 3 within R, R_A and R_B to get that

$$\begin{aligned}
\phi\left(\operatorname{fac}\left(\Delta^I_{ab}\right)\right) &= \min\left(I : a'b'\right) \setminus \left\langle x_1^2, \ldots, x_n^2 \right\rangle, \\
\phi_A\left(\operatorname{fac}\left(\Delta^{I_A}_{a}\right)\right) &= \min\left(I_A : a'\right) \setminus \left\langle x_i^2 \middle| x_i \in A \right\rangle, \\
\phi_B\left(\operatorname{fac}\left(\Delta^{I_B}_{b}\right)\right) &= \min\left(I_B : b'\right) \setminus \left\langle x_i^2 \middle| x_i \in B \right\rangle.
\end{aligned}$$

The first equation then follows from

$$I : a'b' = I_A : a' + I_B : b'.$$

[1]Note to reviewer: I presume that Theorem 7 is known. I am looking into finding a reference for it.

To see how the second equation follows from the first, recall that c is a corner if and only if $\operatorname{lcm}(\phi(\operatorname{fac}(\Delta^I_c))) = \mathbb{x}$. Then we are done as

$$\begin{aligned}
&\operatorname{lcm}\left(\min\left(I : a'b'\right) \setminus \left\{x_1^2, \ldots, x_n^2\right\}\right) = \\
&\quad \operatorname{lcm}\left(\min\left(I_A : a'\right) \setminus \left\langle x_i^2 \middle| x_i \in A \right\rangle\right) * \\
&\quad \operatorname{lcm}\left(\min\left(I_B : b'\right) \setminus \left\langle x_i^2 \middle| x_i \in B \right\rangle\right). \quad \square
\end{aligned}$$

Since content only concerns corners of full support, this allows to compute the content of a slice (I, S, q) from its projections onto R_A and R_B provided A, B are I-independent and S-independent. See [10] for suggestions on what to do if A, B are I-independent but not S-independent.

It is true that independence can never obtain for the initial slice $(I\mathbb{x}, \langle 0 \rangle, 1)$ due to the multiplication by $\mathbb{x}$, but it can obtain for slices considered at a later stage of the algorithm.

9.3 A base case of two variables

If $n = 2$ then the corners and their simplicial Koszul complexes can be computed directly at only the cost of sorting the minimal generators. This can be relevant even if the input ideal is in more than two variables since independence splits generate slices in fewer variables than the input.

Let $\min(I) = \{m_1, \ldots, m_k\}$ where $m_1, \ldots, m_k$ are sorted in ascending lexicographic order with $x_1 > x_2$. There are only two kinds of corners for $n = 2$. The first are the generators $a_1, \ldots, a_k$, and the Koszul simplicial complex for all of these is $\{\emptyset\}$.

The second kind of corner are the maximal staircase monomials. Let $\psi(x^u, x^v) \stackrel{\text{def}}{=} x_1^{v_1} x_2^{u_2}$. Then the maximal staircase monomials are $\psi(a_1, a_2), \ldots, \psi(a_{k-1}, a_k)$. These all have complex $\{\emptyset, \{x_1\}, \{x_2\}\}$.

Example 3. For $I \stackrel{\text{def}}{=} \langle xy^5, x^2y, x^5 \rangle$ we have

$$\begin{aligned}
\operatorname{con}(I\mathbb{x}, \langle 0 \rangle, 1) = \{&(xy^5, \{\emptyset\}), (x^2y, \{\emptyset\}), (x^5, \{\emptyset\}), \\
&(x^2y^5, \{\emptyset, \{x_1\}, \{x_2\}\}), \\
&(x^5y, \{\emptyset, \{x_1\}, \{x_2\}\})\}.
\end{aligned}$$

10. HILBERT POINCARÉ SERIES

In this section we show how to use corners to compute Hilbert-Poincaré series. This is possible because of the well-known and indeed ancient Euler characteristic.

Definition 3. The *Euler characteristic* of Δ is defined by

$$\chi(\Delta) \stackrel{\text{def}}{=} \sum_{v \in \Delta} (-1)^{|v|-1}.$$

It is also well known that the Euler characteristic determines the coefficients of the Hilbert-Poincaré series numerator by way of the upper Koszul simplicial complex. Since non-corners have a zero Euler characteristic, we get that

$$H(I) = \sum_{v \in \mathbb{N}^n} \chi\left(\Delta^I_{x^v}\right) x^v = \sum_{m \in \operatorname{cor}(I)} \chi\left(\Delta^I_m\right) m. \qquad (4)$$

So the knowledge of the corners and their associated Koszul simplicial complexes yields the Hilbert-Poincaré series numerator, and the former is of course precisely what the Slice

Algorithm computes. It only remains to show how to compute the Euler characteristic. [2]

To compute the Euler characteristic, we are going to use a characterization in terms of the square free ideal $\langle\phi(\Delta)\rangle$. Since $\Delta_{\mathbb{x}}^{\langle\phi(\Delta)\rangle} = \Delta$, we get from Equation 4 that

$$\text{Coefficient of } \mathbb{x} \text{ in } H(\langle\phi(\Delta)\rangle) = \chi\left(\Delta_{\mathbb{x}}^{\langle\phi(\Delta)\rangle}\right) = \chi(\Delta).$$

Thus computing the Euler characteristic of a simplicial complex amounts to computing the coefficient of $\mathbb{x}$ in $H(I)$ for I a square free monomial ideal. In this way it makes sense to define $\chi(I)$ as the coefficient of $\mathbb{x}$ in $H(I)$.

We could compute all of $H(I)$ to get $\chi(I)$, but we don't have to. The divide and conquer algorithm by Bigatti et.al. [4, 3] is the best way to compute Hilbert-Poincaré series. It is based on repeated application of the equation

$$H(I) = H(I:p)p + H(I + \langle p\rangle), \tag{5}$$

where p is a monomial. For square free p this implies that

$$\chi(I) = \chi(I:p) + \chi(I + \langle p\rangle),$$

where we embed $I:p$ in the subring of the ambient polynomial ring that excludes those variables that divide p.

If A, B are I-independent sets of variables then it is well known that $H(I) = H(I_A)H(I_B)$, recalling these definitions from Section 9.2. Then we also have $\chi(I) = \chi(I_A)\chi(I_B)$.

In particular, if $m \in \min(I)$ and m is relatively prime to every other minimal generator of I, then

$$\chi(I) = \chi(I')\chi(\langle m\rangle) = -\chi(I'), \quad I' \stackrel{\text{def}}{=} \langle \min(I) \setminus \{m\}\rangle.$$

We can also tell that $\chi(I) = 0$ if $\operatorname{lcm}(\min(I)) \neq \mathbb{x}$, recalling that I is square free.

This suggests a procedure to compute $\chi(I)$. The procedure terminates as long as we choose the monomials p such that $1 \neq p \notin I$, and thus we have an algorithm for computing Euler characteristics. This and the Slice Algorithm for corners then yields an algorithm for Hilbert-Poincaré series.

Running this Euler characteristic algorithm for every corner might seem like it would take a lot of time, but in fact in our implementation it generally takes longer to compute the corners and Koszul simplicial complexes in the first place.

We should point out that this Corner-Euler algorithm for computing Hilbert-Poincaré series is not equivalent to the Bigatti et.al. algorithm, even though the Euler characteristic computation is also based on Equation 5. One way to see this is to consider that there can be corners of $I + \langle p\rangle$ and corners of $I:p$ that are not corners of I. The Bigatti et.al. algorithm can tolerate this because any terms of $H(I:p)p$ and $H(I + \langle p\rangle)$ that correspond to these additional corners will cancel out such that they do not appear in the final output. In contrast the Slice Algorithm looks only for the actual corners.

Since no terms cancel in the output of the Corner-Euler Algorithm, it is possible to output a term and vacate it from memory as soon as it is computed. In contrast the Bigatti et.al. algorithm has to wait for the extra terms to cancel, and so the terms that occur in the Hilbert-Poincaré series numerator are not identifiable until the end of the computation.

[2]Note to reviewer: While the algorithm is expressed in terms of monomial ideals here (which is convenient for the proofs), it can be translated to simplicial complexes by applying ϕ^{-1} to all the equations. As such, I would have thought that this algorithm would be known. However, after a fair amount of looking, I haven't been able to find *any* papers on algorithms for the Euler characteristic of a general *abstract* simplicial complex. I will ask around some more, and I would be grateful for any references.

11. EXPERIMENTS

In this section we gauge the practical performance of our algorithm for corners. Unfortunately, we know of no implementations of algorithms for corners that we might compare ours against. The computation of Hilbert-Poincaré series has, however, received a lot of attention both in the literature and in terms of being implemented, and so we look at the Corner-Euler Algorithm for Hilbert-Poincaré series for this experiment.

We have implemented both the Corner-Euler algorithm and the Bigatti et.al. algorithm in the software system Frobby, which is a system for monomial ideal computations. These implementations are of comparable quality and written by the same person to make the comparison as fair as possible. The implementation of the Bigatti et.al. algorithm in CoCoA [6] is to our knowledge the best implementation out there, so we include that in the comparison as well.

We employ a suite of 4 ideals for the experiment, and we name them respectively generic, nongeneric, squarefree and toric. These ideals have been selected from a long list of possible ideals that we could have used. They have been selected solely on the basis of providing interesting information and for being neither trivial nor so demanding that the experiment will run for a long time. Table 1 has further information.

generic: This ideal has been randomly generated with exponents in the range [0,30000]. The ideal is thus very close to generic.

nongeneric: This ideal has been randomly generated with exponents in the range [0,10]. The ideal is thus far from generic but also far from being square free.

squarefree: This ideal has been randomly generated with exponents in the range [0,1]. It is thus square free and farthest from generic.

toric: This ideal is the initial ideal of a toric ideal defined by a primitive vector with eight entries that are random numbers of 30 decimal digits each. The ideal is generic and has exponents in the range [0,95998]. Computing the Hilbert-Poincaré series of this ideal is a subalgorithm in computing the genus of the numerical semigroup generated by the primitive vector.

We run two experiments, one for computing the $\mathbb{N}^n$-graded Hilbert-Poincaré series, and the other for the conventional total degree-graded Hilbert-Poincaré series. These are shown in Table 2 and 3 respectively.[3]

One conclusion we can draw is that the Corner-Euler algorithm is faster for the multigraded computation than for

[3]Note to reviewer: We have been unable to obtain times for CoCoA for the $\mathbb{N}^n$-graded Hilbert-Poincaré series seemingly due to a bug in CoCoA. We are in contact with the authors of CoCoA to remedy this. Also, there is an anomaly in the data on the input nongeneric, where CoCoA is drastically faster than Frobby even using the same algorithm. We are investigating the reason for this.

name	n	$\lvert\min(I)\rvert$	terms of $H(I)$	corners
generic	10	160	2,940,226	
nongeneric	10	200	796,931	
squarefree	20	4,000	251,650	
toric	8	2,099	2,948,154	

(the number of corners will be in the final version)

Table 1: Further information about the ideals.

software algorithm	Frobby Corner-E.	Frobby Bigatti ea.	CoCoA4 Bigatti ea.
generic	59s	1,467s	
nongeneric	35s	73s	
squarefree	156s	60s	
toric	65s	262s	

Table 2: Multigraded Hilbert-Poincaré series.

the univariate one. This is because the Corner-Euler algorithm can output terms as soon as they are computed in the former case, but in the latter case it is necessary to collect like terms before output and this takes extra time.

The Bigatti et.al. algorithm has the opposite behavior, being faster for the univariate computation than the multivariate one. This is because univariate computations allow a base case that is very fast when the degrees of the generators are not too high. Otherwise the base case is exponential in the number of variables, and avoiding this is part of the benefit that the Bigatti et.al. algorithm derives from the univariate computation.

We conjecture that the reason that the Corner-Euler algorithm is faster than the Bigatti et.al. algorithm for generic ideals is that in those cases the number of terms that the Bigatti et.al. algorithm generates that are not actually part of the output is much higher than for other ideals.[4]

12. REFERENCES

[1] Dave Bayer and Mike Stillman. Computation of hilbert functions. *Journal of Symbolic Computation*, 14(1):31–50, 1992.

[2] Dave Bayer and Amelia Taylor. Reverse search for monomial ideals. *Journal of Symbolic Computation*, 44:1477–1486, 2009.

[3] Anna M. Bigatti. Computation of Hilbert-Poincaré series. *Journal of Pure and Applied Algebra*, 119(3):237–253, 1997.

[4] Anna Maria Bigatti, Pasqualina Conti, Lorenzo Robbiano, and Carlo Traverso. A "divide and conquer" algorithm for Hilbert-Poincaré series, multiplicity and dimension of monomial ideals. In *Applied algebra, algebraic algorithms and error-correcting codes (San Juan, PR, 1993)*, volume 673 of *Lecture Notes in Comput. Sci.*, pages 76–88. Springer, Berlin, 1993.

[5] Anna Maria Bigatti and Eduardo Saenz de Cabezon. (n-1)-st koszul homology and the structure of monomial ideals. arXiv:0811.1013v1.

[6] CoCoATeam. CoCoA: a system for doing Computations in Commutative Algebra. Available at `http://cocoa.dima.unige.it`.

[7] Eduardo Saenz de Cabezon. Combinatorial koszul homology: Computations and applications, 2008. `http://arxiv.org/abs/0803.0421`.

[8] Ezra Miller and Bernd Sturmfels. *Combinatorial Commutative Algebra*, volume 227 of *Graduate Texts in Mathematics*. Springer, 2005.

[9] Ezra Miller, Bernd Sturmfels, and Kohji Yanagawa. Generic and cogeneric monomial ideals. *Journal of Symbolic Computation*, 29(4-5):691–708, 2000. Available at `http://www.math.umn.edu/~ezra/papers.html`.

[10] Bjarke Hammersholt Roune. The slice algorithm for irreducible decomposition of monomial ideals. *Journal of Symbolic Computation*, 44(4):358–381, April 2009.

[4]Note to reviewer: We are looking into counting the number of superfluous terms generated by the Bigatti et.al. algorithm to test this hypothesis. Also, there is an unpublished technique for speeding up the Bigatti et.al. algorithm for generic ideals. It is not used by CoCoA but we are looking into including it for the benchmark of the Bigatti et.al. in Frobby.

software algorithm	Frobby Corner-E.	Frobby Bigatti ea.	CoCoA4 Bigatti ea.
generic	68s	1,359s	1,811s
nongeneric	37s	37s	<1s
squarefree	156s	2s	4s
toric	73s	233s	531s

Table 3: Univariate Hilbert-Poincaré series.