On Vector Enumeration

S. A. Linton
Queen Mary and Westfield College
Mile End Road
London E1 4NS, England

Submitted by Gerhard O. Michler

ABSTRACT

We develop further the linear Todd-Coxeter algorithm described previously. In particular, by slightly extending the algorithm, we obtain a simpler description of the calculation which it performs. We also describe an application of this method to a problem in computational representation theory, which itself has applications in algebraic topology and possibly other areas.

INTRODUCTION

In [3], a linear version of the well-known Todd-Coxeter algorithm is described. The same algorithm was developed independently in [1], from the viewpoint of solving systems of polynomial equations in noncommuting variables. It is from there that we take the name "vector enumeration" for these techniques.

The algorithm and the author's implementations of it have been applied in a number of areas, though unfortunately the work is too recent for references to be available. The applications include the construction of soluble quotients of finitely presented groups; analysis of the structure of permutation modules for certain groups in characteristic dividing the group order; the modular representation theory of the Hecke algebras of the exceptional Lie algebras; geometrical analysis of certain irreducible representations of sporadic simple groups; and as a possible alternative to Gröbner basis methods for certain problems in computational algebraic geometry.

In [3], the algorithm is presented as a method of constructing matrix representations of finitely presented groups (or equivalently of their group

655 Avenue of the Americas, New York, NY 10010 0024-3795/93/$6.00

algebras). In Section 1 of this paper, we recast the algorithm as a means of constructing representations of arbitrary finitely presented algebras. This subsumes the previous setting, but is easier to describe, suggesting that it is more natural.

The algorithm is briefly recounted in Section 2, as it appears in its new setting. For details and notes on implementation, the reader is referred to [3].

Given a group G, with a matrix representation on a vector space V, and (vectors generating) a G-invariant subspace $U \leqslant V$, the action of G on V/U can be constructed. The usual method of performing this construction (see [5]) involves first constructing an echelonized basis for the subspace U, and then constructing the action on V/U in terms of the nonpivot columns of this basis.

In Section 3, we describe an alternative technique, based on vector enumeration, which avoids constructing an explicit basis for U. This is much more efficient in the case when $\dim V/U \ll \dim U$ and various vectors and matrices are sparse. These conditions occur, for example, when computing the homology groups of distance-transitive graphs, and this technique has been used to good effect for that purpose.

1. THE NEW FORMULATION

The algorithm of [3] can be trivially modified to produce an algorithm meeting the following specification: The input to the algorithm consists of k, a field (subject to certain restrictions, discussed below), and X, a finite set of symbols.

We can regard X and k as specifying an algebra, namely the free (associative, but not commutative) k-algebra generated by X. We denote this by A. Every element of A may be written as an expression involving elements of X and of k, combined by addition and multiplication, and the next section of our input will be made up of such expressions, which we regard as elements of A. Specifically, it consists of a finite set $R \subseteq A$ of relators.

The set R must generate a two-sided ideal $I = \langle ARA \rangle$, and the quotient of A by this ideal will itself be a k-algebra, which we call P, with a natural quotient map $q : A \to P$.

This quotient P will be generated by $q(X)$, and $q(R)$ will be 0 in P. We can recast the latter statement by regarding our relators once again as expressions in X and k. The statement that $q(R) = 0$ then tells us that when we evaluate any of these expressions with $x + I$ in place of x, for each x in X (so that we are computing in P), we must obtain 0. Furthermore, it is easy to see that P is the largest possible (indeed, the universal) k-algebra, generated by a set in bijection with X, such that the formulae R, evaluated at

the images of X, all give 0 in the algebra. Thus it is clearly correct to describe P as the algebra presented by the generators X and the relators R.

Our algorithm will, if it succeeds, construct a matrix representation of this algebra P in its action on a certain P-module M. The remainder of the input is the specification of this module. Now the algebra P acts on itself by right multiplication, and it is easy to see that P, under this action, is a right P-module. Furthermore, for any positive integer s, the Cartesian power P^s is also a right P-module and any s-generator right P-module must be a quotient of it. Accordingly, the final part of the input consists of a positive integer s and a set W of s-tuples of expressions in X and k. We view W as a subset of A^s and call its members submodule generators.

We define the homomorphism $q^s : A^s \mapsto P^s$ by $(a_1, \ldots, a_s)q^s = (a_1 + I, \ldots, a_s + I)$. The images of the submodule generators W under this map lie in P^s. We denote by N the submodule $\langle Wq^sP^s \rangle$ of P^s which they generate. Our algorithm constructs the action of P on the quotient module $M = P^s/N$.

Specifically, if M is finite dimensional, our algorithm will terminate, having constructed matrices giving the action of the generators Xq on M. More pedantically, the matrices give an action of Xq on a k-vector space V which is P-module isomorphic to M. The actual isomorphism can be computed at negligible extra cost, giving a representative in A^s (of which M is a quotient), for each element of the basis of the vector space, and giving the vector in V corresponding to the generating elements $(1, 0, \ldots, 0)$, $(0, 1, 0, \ldots, 0)$, etc. of A^s.

EXAMPLE 1 (A permutation module). In practice, it is always better to use the classical Todd-Coxeter algorithm to construct permutation representations of groups, and modifications of it exist for semigroups and monoids. However, as was proved in [3], the linear algorithm can perform any of these calculations.

We know that the group with presentation $\langle a, b \mid a^4 = b^2 = (ab)^2 = 1 \rangle$ is the dihedral group of order 8. We can construct its permutation module of degree 4, over any field k, by inputting k, the generators $X = \{a, b, a', b'\}$, and the relators $R = \{aa' - 1, a'a - 1, bb' - 1, b'b - 1, a^4 - 1, b^2 - 1, (ab)^2 - 1\}$. This module is cyclic, as the permutation representation is transitive, so we may set $s = 1$. Finally, we select our module by setting $W = \{b - 1\}$.

Applying the algorithm to this case, we obtain the matrices

$$a = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \end{pmatrix}, \qquad b = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

and their inverses for a' and b'. We also obtain the information that the (only) module generator has image $(1, 0, 0, 0)$ in the vector space, while the four basis vectors are images of 1, a', $a'b$, and a'^2.

EXAMPLE 2 (A quotient of a permutation module). Like all permutation modules, this one fixes the all-ones vector $(1, 1, 1, 1)$, which we know to be an image of $1 + a'(1 + b + a')$. We can construct the quotient of the permutation module by this one-dimensional submodule by adding the word to W. We then obtain the matrices

$$a = \begin{pmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ -1 & -1 & -1 \end{pmatrix}, \qquad b = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix}.$$

EXAMPLE 3 (A noncyclic module). A permutation module of a group ring is cyclic (that is, generated as a module by one element) just when the permutation representation is transitive. An intransitive permutation representation can be easily constructed from its transitive components, but in general it is not so easy to construct an arbitrary module from cyclic submodules. Accordingly, it can be worthwhile to construct noncyclic modules directly.

As an example we take two copies of the representation constructed in Example 1, and fuse their one-dimensional submodules. The generators and relators are as before, and now $s = 2$ and $W \subseteq A^2 = \{(b - 1, 0), (0, b - 1), (1 + a'(1 + a' + b), -1 - a'(1 + a' + b))\}$. We obtain a representation of degree 7, given by

$$a = \begin{pmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & -1 & 1 & 1 & 1 & -1 & -1 \end{pmatrix},$$

$$b = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix}.$$

The two module generators are $(1,0,0,0,0,0,0)$ and $(0,1,0,0,0,0,0)$, while the seven basis vectors are images of $(1,0)$, $(0,1)$, $(a',0)$, $(a'b,0)$, $(a'^2,0)$, $(0,a')$, and $(0,a'b)$.

EXAMPLE 4 (The symmetric inverse monoid). This is the monoid of all partial bijections on a given set. For a set of size four, it is given by the presentation

$$\langle s_1, s_2, s_3, e \mid s_i^2 = (s_1s_2)^3 = (s_2s_3)^3 = (s_1s_3)^2 = 1,$$

$$e^2 = e,\ s_1e = es_1,\ s_2e = es_2,\ es_3e = (es_3)^2 = (s_3e)^2\rangle.$$

This presentation, converted to an algebra presentation by replacing each relation $a = b$ with the relator $a - b$, gives the regular representation of the monoid algebra, which is of degree 209. The monoid has, however, a natural representation of degree 4 as partial permutation matrices, and we obtain this by setting $W = \{e, s_1 - 1, s_2 - 1, s_3e - s_3\}$, obtaining matrices

$$s_1 = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}, \qquad s_2 = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$s_3 = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}, \qquad e = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$

EXAMPLE 5 (The Hecke algebra of type A_3). There is a Hecke algebra associated with any Coxeter group, generated by elements corresponding to the fundamental roots of the group, and with a presentation derived from the coxeter presentation of the group. For full details see [2].

The Hecke algebra is normally constructed over the ring $\mathbb{Z}[q, q^{-1}]$ of Laurent polynomials, but by reducing mod 3 and evaluating at $q = -1$ we obtain the following representation for the Hecke algebra of type A_3 over GF(3):

$$\langle x, y, z \mid x^2 - x + 1,\ y^2 - y + 1,\ z^2 - z + 1,$$

$$yxy - xyx,\ zyz - yzy,\ zx - xz\rangle.$$

It has a natural representation of the same dimension as the Lie algebra of type A_3, namely 4. This representation can be obtained by setting $W = \{x + 1, y + 1\}$. This gives the output

$$x = \begin{pmatrix} -1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & -1 & 1 \end{pmatrix}, \quad y = \begin{pmatrix} -1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & -1 & 1 & 0 \\ 0 & 0 & 0 & -1 \end{pmatrix},$$

$$z = \begin{pmatrix} 0 & 1 & 0 & 0 \\ -1 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & -1 \end{pmatrix}.$$

All of these examples are, of course, very small representations, but in fact the author's programs have no problems with arbitrary representations of degree several hundred, such as the Hecke algebra of type E_8, or sparse representations (such as permutation representations) of degree several thousand.

Possible Fields

The algorithm itself imposes no restriction on the field k, except that exact calculation must be possible. While this appears to rule out $\mathbb{R}$ and $\mathbb{C}$, it may still be possible to proceed, provided that the coefficients in R and W are exact. They will then generate an extension of $\mathbb{Q}$ of finite transcendence degree, in which exact calculation would be possible.

That said, the present implementation by the author is designed mainly for finite fields of small prime order (though other finite fields would not be a problem). A version working in $\mathbb{Q}$ exists, but no attempt has yet been made to deal with the problem of coefficient explosion.

There is reason to believe that this problem may be severe, as the coincidencing process in vector enumeration is essentially Gaussian elimination, and that is well known to give a severe explosion problem in exact calculation over fields of characteristic 0. The implementation of [1] (in LISP) appears to operate over such fields, but the examples described are of relatively small degree, and do not appear to involve extensive collapse. There are known techniques for reducing the coefficient explosion in Gaussian elimination, but it is not yet clear whether they can be applied in this situation.

Finally, it would be very convenient to be able to perform these calculations not just over fields, but over Euclidean domains (in particular over $\mathbb{Z}$).

The present algorithm does not extend to these cases, as there is the possibility of obtaining information such as $2b = 3b'$, from the relators (the notation is defined below). This must be retained, but cannot be used to eliminate any basis vector. This problem is the subject of current research, based on using lattice basis reduction and Smith normal form techniques to manipulate this extra information. An experimental program now exists.

2. THE ALGORITHM IN BRIEF

In this section we give a brief outline of the algorithm, in terms of the formulation above. This concentrates on the mathematical basis, rather than the practicalities, for which see [3].

The Table

The algorithm aims to deduce, from the relators R and submodule generators W, the complete structure of the module M and the action of P on M. As the calculation proceeds, the information obtained so far is represented in a table. The rows of this table are indexed by an ordered set B, taken from a universal set $\mathscr{B}$ (such as the nonnegative integers). Many of the entries of the table will lie in a vector space with basis identified with B, which we call kB. We will also use kS for the subspace of kB spanned by S, for various subsets S of B. Entries may also take a value $\perp$, denoting empty or unknown. The b row of the table consists of the following parts: a flag $d_b \in \{\textbf{true}, \textbf{false}\}$; an entry $r_b \in kB \cup \perp$; an element p_b of A^s; and a set of entries $v_{b,x} \in kB \cup \perp$, one for each $x \in X$.

We divide the rows of the table into two subsets $B^u = \{b \in B \mid d_b = \textbf{false}\}$ and $B^d = \{b \in B \mid d_b = \textbf{true}\}$. We call B^u the set of *undeleted* rows and B^d the set of *deleted* rows.

The interpretation of the table depends on establishing a linear map ϕ between kB and A^s. This is given on B by $\phi(b) = p_b$ and extended linearly to kB. It can, of course be composed with q^s and the quotient homomorphism of P^s onto M, to yield a linear map ψ from kB to M. Initially, the map ψ is neither one-one nor onto, and as kB has no A-module structure, it cannot be a homomorphism; but we will show that, if the algorithm completes, ψ will provide an A-module isomorphism between kB^u and M.

We impose a number of conditions on the table. As we work with it, we have to arrange our manipulations so that the table continues to meet these conditions. The conditions on a row depend on whether it lies in B^u or B^d.

For each $b \in B^u$ we require that $r_b = \perp$, and that for each $x \in X$, either $v_{b,x} = \perp$ or $v_{b,x} \in kB^u$, and $\psi(b)x = \psi(v_{b,x})$. For each $b \in B^d$ we

require that $r_b \in k\{b' \mid b' < b\}$ and that $\psi(b) = \psi(r_b)$. We say that row b has been deleted and replaced by r_b.

For any vector $v \in kB$, we define $u(v)$, the undeleted image of v, as follows. Suppose that $v = \sum_{i=1}^{m} \lambda_i b_i$; then

$$u(v) = \sum_{i=1}^{m} \lambda_i \begin{cases} b_i & \text{if } b_i \in B^u, \\ u(r_{b_i}) & \text{otherwise.} \end{cases}$$

Since $r_b \in k\{b' \mid b' < b\}$ the recursion implied in the above definition must terminate.

The algorithm proceeds by modifying the table. New elements are added to B to represent additional elements of A^s, and the words in R and W are used to deduce linear dependencies among the images $\psi(B)$, which are used to move elements from B^u to B^d.

Applying the Relators

The entries $v_{b,x}$ provide a partial action for X on kB^u via

$$b.x = \begin{cases} v_{b,x} & \text{if } v_{b,x} \neq \perp, \\ \text{undefined} & \text{otherwise.} \end{cases}$$

This can be extended naturally to a partial action of A on kB^u. We call this the action "without defining."

We can extend this to a "total action with side effects" by adding an element of $\mathscr{B} \setminus B$ to B whenever necessary. Furthermore, we can do this compatibly with the restrictions above. Specifically, to apply this action to $b \in B^u$ and $x \in X$:

```
if v_{b,x} ≠ ⊥
    then return v_{b,x}
    else
        take b' from 𝔅 \ B such that b' > B
        add b' to B
        set p_{b'} := p_i b x
        set v_{b,x} := b'
        set v_{b',x} := ⊥ for all x ∈ X
        set d_{b'} := false
        set r_{b'} := ⊥
        return b'
fi
```

This procedure can be extended to give an action (with side effects) of A on kB^u.

The basic step in the calculation then consists of taking an equation known to be true in the action of A on M, and computing its equivalent in the action with defining of A on kB^u. If this equivalent is true, then nothing need be done; if not, then we obtain a linear relation among the members of B^u, which we call a *coincidence*.

Processing Coincidences

A coincidence is a vector $v = \sum_{i=1}^{m} \lambda_i b_i$ such that $\psi(v) = 0$. Having discovered such a vector, we wish to apply this information to the table. As we shall see, this application may lead us to discover further coincidences. In order to avoid the complication of applying more than one coincidence at a time, it is convenient to place such newly discovered information on a stack, and apply it later. This does however mean that, while the vector v must have been a member of kB^u when discovered, it can now only be assumed to lie in kB. Accordingly, we give a procedure for applying the information that $\psi(v) = 0$ for $v \in kB$:

```
replace v by u(v)
if v = 0 then stop
set b := max_{i=1}^{m}(b_i)
set l such that b = b_l
set v_0 := -1/λ_l(v - λ_l b_l)
for x ∈ X do
    if v_{b,x} ≠ ⊥ then
        set v' to be the image with defining of v_0 under x
        stack the coincidence that ψ(v_{b,x} - v') = 0
      set v_{b,x} := ⊥
    fi
od
set r_b := v_0
set d_b := true
for every b' and x such that v_{b',x} ≠ ⊥ and v_{b',x} has nonzero
    coefficient of b do
    replace v_{b,x} by u(v_{b,x}).
od
```

In practice a much more complex procedure is used, for efficiency, but mathematically it is equivalent to the above.

One can verify that, after the above procedure has been carried out, all the conditions on the table imposed above are still satisfied; that $u(v)$ is now 0; and that any of the equations that had already been applied remain true.

Starting and Finishing Conditions; Sketch Proof of Correctness

We start the algorithm with just s elements $b^{(1)}, \ldots, b^{(s)}$ in B, and with $d_{b(i)} = \textbf{false}$, $r_{b(i)} = v_{b^{(i)},x} = \perp$, and $p_{b^{(i)}} = (0, \ldots, 0, 1, 0, \ldots, 0)$ (the 1 is in the ith place) for $1 \leqslant i \leqslant s$ and $x \in X$. This table certainly meets the requirements above.

We then consider equations of three types. Firstly, for each $b \in B^u$ and $r \in R$ we look at $p_b r = 0$. Secondly, for each $w = (w_1, \ldots, w_s) \in W$ we use $p_{b^{(1)}} w_1 + \cdots + p_{b^{(s)}} w_s = 0$. Finally, for each $b \in B^u$ and $x \in X$, we consider the trivial equation $p_b x = p_b x$, so as to ensure that the table entry $v_{b,x}$ does in fact get filled in (otherwise the algorithm would terminate immediately when given the free algebra).

Whenever there is no $b \in B^u$ and $x \in X$ such that $v_{b,x} = \perp$, we say the table is *closed*. In this situation the partial action without defining becomes a total action of A on kB^u. This occurs just when all the equations of the third form above are true in kB^u. In this situation, we can define an A-module homomorphism θ from A^s to kB^u by letting $\theta(p_{b^{(i)}}) = u(b^{(i)})$. Since the $p_{b^{(i)}}$ generate A^s, this is enough to define θ.

It is easy to see that if, additionally, all the equations of the first type hold, then $\ker q^s \leqslant \ker \theta$, so that θ factors through q^s and gives a map from P^s to kB^u. Similarly the equations of the second type imply that $N \leqslant \ker \theta$, showing that θ may be regarded as a P-module homomorphism from M to kB^u.

However, regarding θ in this light, we can show that it is an isomorphism, since the map ψ defined above provides an inverse. This shows that, should the table reach a closed form in which all the equations defined above are satisfied, then kB_u will be P-module isomorphic to M and the algorithm will have succeeded.

There is, of course, no guarantee that the table will ever reach such a closed form, as M might be infinite dimensional. If M is finite dimensional, then, provided the equations are processed in an order meeting certain conditions, it is possible to show that the algorithm must complete. The proof is very similar to the equivalent result for the Todd-Coxeter algorithm (see, for example, [4]).

EXAMPLE. We take the algebra $\langle x \mid 3x^2 - 2x - 1 \rangle$, and we shall work over $\mathbb{Q}$. We let $s = 1$ and $W = \varnothing$. The initial table is thus

b	p_b	d_b	r_b	$v_{b,x}$
$b^{(1)}$	1	**false**	$\perp$	$\perp$

We begin by applying the relator to the only row we have, row $b^{(1)}$. That is we check the equation $p_{b^{(1)}}.(3x^2 - 2x - 1) = 0$, which is certainly true in

the module M we aim to construct, to see if its equivalent is true of our table. In order to do this we compute the image under the action with defining of $3x^2 - 2x - 1$ on $b^{(1)}$. This computation begins by looking at $v_{b^{(1)},x}$, which is $\perp$, so according to the first procedure above, we add a new element to B, which we call b_2. The procedure tells us how to modify the $b^{(1)}$ row and how to set up the b_2 row. We obtain

b	p_b	d_b	r_b	$v_{b,x}$
$b^{(1)}$	1	**false**	$\perp$	b_2
b_2	x	**false**	$\perp$	$\perp$

We find that in order to evaluate $b^{(1)}.(3x^2 - 2x - 1)$ we have to define another row, say b_3, and we obtain

b	p_b	d_b	r_b	$v_{b,x}$
$b^{(1)}$	1	**false**	$\perp$	b_2
b_2	x	**false**	$\perp$	b_3
b_3	x^2	**false**	$\perp$	$\perp$

We can then read off that $b^{(1)}.(3x^2 - 2x - 1) = -b^{(1)} - 2b_2 + 3b_3$. Since this must be 0 in M, we obtain the coincidence that $\psi(-b^{(1)} - 2b_2 + 3b_3) = 0$. Following the second procedure above, we delete b_3 and obtain

b	p_b	d_b	r_b	$v_{b,x}$
$b^{(1)}$	1	**false**	$\perp$	b_2
b_2	x	**false**	$\perp$	$\dfrac{2b^{(1)} + b_2}{3}$
b_3	x^2	**true**	$\dfrac{2b^{(1)} + b_2}{3}$	$\perp$

Notice how the expression $(2b^{(1)} + b_2)/3$ has been substituted for b_3 in the entry for $v_{b_2,x}$, so that the first two rows of the table no longer refer to the third.

We can observe that this table is closed, so that no further definitions can be needed. It remains to apply the equation $p_{b_2}.(3x^2 - 2x - 1) = 0$. When we compute $b_2.(3x^2 - 2x - 1)$ we find that it is already 0, so that we are finished. We conclude that the action of x on this module (the regular representation of the algebra, in fact) is given by the matrix

$$x = \begin{pmatrix} 0 & 1 \\ \frac{2}{3} & \frac{1}{3} \end{pmatrix},$$

where the generator of the module is the vector (1, 0), while the basis is the images of 1 and x.

3. CONSTRUCTING QUOTIENT ACTIONS

Consider a finitely generated group $G = \langle X \rangle$ with a matrix representation $\rho : G \to \mathrm{GL}(V)$, where V is a k-vector space and $\dim V = n$. Suppose that there is a G-invariant subspace $U < V$; then G will act on the quotient space V/U.

Given G in the form of matrices $\rho(X)$ (and so, implicitly, given V, n, and an ordered basis $B = \{b_1, \ldots, b_n\}$ for V), and given vectors in V whose images under $\rho(G)$ span U, we can construct the matrix action of G on V/U.

The traditional methods of doing this, for example in the SPLIT program of the Meat-Axe [5], involve constructing an echelonized basis of U with respect to the basis B of V. When $\dim U$ is not much smaller than n, so that $\dim(V/U)$ is small, most of the calculations are in fact occurring in U, even when the action on V/U is all that is to be found. We will see that the vector enumeration algorithm can be used to circumvent these calculations under suitable conditions. Such conditions definitely occur in the computation of graph homology and they may well arise elsewhere.

We can convert this problem into a vector enumeration problem as follows. Let the subspace U be given by vectors $\{u_1, \ldots, u_m\}$. The field k is given, and we take as many generators as we have generators of G and no relators. We take the number of module generators s to be n. Since k can be embedded naturally in A, we can rewrite any vector $v \in V \cong k^n$ as a vector in $\bar{v} \in A^s = A^n$. If we take submodule generators

$$W_1 = \{\bar{b}x - \overline{b\rho(x)} \mid b \in B,\ x \in X\},$$

then it is easy to see that the matrices we will obtain will just be $\rho(X)$. When we adjoin the set

$$W_2 = \{\bar{u}_i \mid 1 \leqslant i \leqslant m\}$$

to W_1, we obtain a quotient of this representation by the submodule generated by W_2, which is exactly $\bar{U}$, so that we obtain the desired action.

EXAMPLE. We consider the deleted permutation representation of A_6, of degree five. This is given over $\mathbb{F}_2$ by two generating matrices:

$$\begin{pmatrix} 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}, \quad \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 \end{pmatrix}.$$

We can convert these matrices into a presentation for a five-generator module for the free $\mathbb{F}_2$-algebra on two generators with ten submodule generators corresponding to the rows of the above matrices. This presents a five-dimensional module on which the generators of the algebra act as the above matrices. Writing x and y for the algebra generators, the complete list of submodule generators is

$$\begin{array}{lllll} (x,0,1,0,0) & (1,x,0,0,0) & (0,1,x,0,0) & (0,0,0,x+1,0) & (0,0,0,0,x+1) \\ (y+1,0,0,0,0) & (0,y,0,0,1) & (0,1,y,0,0) & (0,0,1,y,0) & (1,1,1,1,1+y) \end{array}$$

Feeding this presentation to the vector enumerator will simply return the matrices with which we started; however, these matrices fix a four-dimensional subspace (corresponding, in the full permutation representation, to the vectors of weight 0) which is spanned by images of vectors such as (1, 1, 0, 0, 0). Accordingly, if we take as several such vectors as additional submodule generators, we obtain a one-dimensional quotient of the five-dimensional module, together with the (trivial) action of the generators on the quotient and the images under the quotient map of the five module generators.

Practical Considerations

The presentations and submodule generators used in these calculations are very different from those encountered in other applications of vector enumeration, and somewhat different strategies are needed to obtain the best results.

In particular, the submodule generators W_2 corresponding to U should be applied before those in W_1 which enforce the action on V. The aim of this is to reduce the dimension as quickly as possible to something close to $\dim(V/U)$, where the remaining submodule generators can be checked more quickly.

Another major consideration is sparseness. While the dimension remains large, it is important that most of the vectors being manipulated remain sparse, since otherwise the calculations reduce to Gaussian elimination of

large matrices, which is what we are trying to avoid. The key to achieving this seems to be to use many redundant (sparse) generators for U. This keeps the chains of coincidences and consequences relatively short for the most part, which has the desired effect.

In practice these calculations are performed not with a general-purpose vector enumeration program, but with a separate program that reads a special input format and applies the submodule generators directly to the table. In the language of [3], each element of W_1 provides a deduction, while each element of W_2 gives an equation.

This program has been used with success to compute quotient actions of dimension 10 or so of modules of dimension as large as 5,000, and there seems no reason to believe that substantially larger problems could not be tackled in the same way.

REFERENCES

1 Gilles Labonté, An algorithm for the construction of matrix representations for finitely presented non-commutative algebras, *J. Symbolic Comput.* 9:27–38 (1990).

2 J. E. Humphreys, *Reflection Groups and Coxeter Groups*, Cambridge Stud. Adv. Math. 29, Cambridge U.P., 1990.

3 S. A. Linton, Constructing matrix representations of finitely presented groups, *J. Symbolic Comput.* 12:427–438 (1991).

4 J. Neubüser. An elementary introduction to coset table methods in computational group theory, in *Groups—St. Andrews 1981* (C. M. Campbell and E. F. Robertson, Eds.), Cambridge U.P., 1982, pp. 1–45.

5 R. A. Parker, The computer calculation of modular characters (the Meat-Axe), in *Computational Group Theory* (Michael Atkinson, Ed.), Academic, London, 1984, pp. 267–274.

Received 4 November 1992; final manuscript accepted 14 May 1993