J. Symbolic Computation (1991) **12**, 415-426

Double Coset Enumeration

STEPHEN A. LINTON

Department of Pure Mathematics and Mathematical Statistics,
16 *Mill Lane, Cambridge CB*2 1*SB, UK*

(*Received* 9 *May* 1988)

Coset enumeration is the principal method for solving the word problem in finitely presented groups. The technique has a long history and was one of the first applications of electronic computers to pure mathematics. Present applications are limited by the space available to store the coset table in computer memory. Enumerating double cosets offers a substantial saving of space in suitable cases. An algorithm is described with some notes on implementation.

1. Introduction

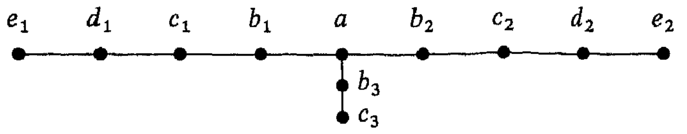
In order to work with a finitely presented group, it is normally necessary to devise a means of determining if a given word in the generators is equal to the identity element of the group, or equivalently to determine whether two words generate the same element. Such a means is called a solution for the *word problem* of the group. It is widely known that the word problem for finitely presented groups is unsolvable in general (see, for example, chapter 13 of Rotman, 1984). However, in a great many cases of practical interest (such as groups of finite order) it can be solved, and a major technique for doing so is *coset enumeration.* This technique has a long history, dating back to the 1930s, and was one of the first areas of pure mathematics to make use of electronic computers when they became available. An account of the basic techniques appears in Leech (1984) and the history is described in Leech (1963). The principal limitation on modern computer implementations of (single) coset enumeration is the space required to store the coset table. An enticing approach to circumventing this problem is to enumerate double, rather than single, cosets. This paper describes an algorithm for double coset enumeration and includes some notes on computer implementation.

The basis of the algorithm was described by Conway in 1984. A more detailed description, correcting some errors and omissions was provided by Parker (1987), and some further errors and omissions are corrected here and it is implemented. The algorithm has been used to check a presentation for $2^2.{}^2E_6(2)$, whose largest subgroup has index 3 968 055, that could not be handled by existing (Single Coset Enumeration) programs.

The basic situation is as follows: we have a finitely presented group G with two subgroups, H—which is considered to be large, typically a maximal subgroup, and K—which is considered to be a small well-understood subgroup. We enumerate the double cosets $HgK \subseteq G$ and establish the action of G upon them. The algorithm as described imposes very severe restrictions on the choice of subgroup K; in section 6 a number of possible methods for relaxing these restrictions are discussed.

Essentially, the process is equivalent to the enumeration of single cosets Hg, using knowledge of K to handle all the cosets in a K-orbit at once for most purposes, thereby saving space and time. In particular, the permutation action of G on the single cosets Hg can readily be obtained from the output of the Double Coset Enumerator.

A reasonable example to illustrate the process is the sporadic group Fi_{24}. This is presented as a Coxeter group (see ATLAS, p. 232) Y_{442} as shown below, with some additional relations.



We take

$$H = Y_{432} = \langle a, b_1, c_1, d_1, b_2, c_2, d_2, e_2, b_3, c_3 \rangle \cong Fi_{23} \times 2.$$

G has 306 936 single cosets Hg which have, in fact, been enumerated as such. We can take

$$K = \langle b_1, c_1, d_1, e_1 \rangle \cong S_5,$$

whereupon the single cosets are gathered into 6332 double cosets HgK. We can see at once that almost all the generators of G, apart from those contained in K, commute with K, which, as we shall see later, is the basic requirement for the algorithm to be useful.

This and other examples throughout the paper are drawn from the conjecture of Conway, Norton, Soicher and others concerning a possible presentation for the Monster group. The conjecture, and various related results which have been proved, are described in Conway *et al.* (1985*b*) and Conway & Pritchard (1986).

We shall start by describing the particular model for single coset enumeration which we use and establishing some notation. Once single coset enumeration is seen in the right light it is easy to see how understanding of K can be used to extend this to double coset enumeration in a fairly natural way.

Note: The notation of the ATLAS is used without comment throughout.

2. Single Coset Enumeration

2.1. Introduction

This section describes the particular technique of single coset enumeration which we will later extend to double cosets. For a survey of various techniques see Cannon *et al.* (1973).

We shall consider a finitely presented group $G = \langle X \mid R \rangle$, where X is an inverse-closed generating set and the relator xx^{-1} has been adjoined to R for each $x \in X$. We also have a set W of words in X, which generate the subgroup H.

2.2. The Single Coset Table

Following the Todd–Coxeter algorithm we maintain a set S of "single cosets" and a table which can be considered as a function $f: S \times X \to S \cup \{0\}$. We read $f(s, x) = s' \neq 0$ as meaning that $sx = s'$ in the "action" of G on the set of cosets S and $f(s, x) = 0$ as meaning that sx is still unknown.

We have a *Consistency Condition* on our table that

$$f(s, x) = s' \neq 0 \Leftrightarrow f(s', x^{-1}) = s,$$

so that inverses have the behaviour we expect. We start with an empty table ($S = \{s\}$ and $f(s, x) = 0 \ \forall \ x \in X$) and progressively modify it, retaining the consistency condition, until it represents the permutation action of X on the cosets Hg. In terms of the table this means that $f(s, x) \neq 0 \ \forall s, x$, that the words in R fix all $s \in S$ and that the words in W fix some coset s_1 which we identify with H. We also have to ensure that cosets which are distinct in G remain distinct in S.

It is also convenient to have some way of recording in the table when a coset s has been proved to be the same (in G) as an earlier coset s' so that references to s can be diverted to s'. We say that coset s has been *deleted*. The *undeleted image* of s is s', if s' has not been deleted or the undeleted image of s' if it has. Since s' is always earlier in the table than s this chain must terminate. From time to time we *pack* the table by replacing all references to deleted cosets with references to their undeleted images, and then reclaiming the space that was occupied with deleted cosets.

2.3. PUSHING RELATORS

We follow the HLT version of the Todd–Coxeter algorithm (see Leech, 1984) using various refinements. In this process the basic operation is *pushing a relator-coset pair* (r, s). The effect of this is to ensure, by modifications to the table, that if $r = x_1x_2 \ldots x_n$ then $s^{(0)} = s$, $s^{(1)} = s^{(0)}x_1$, $s^{(2)} = s^{(1)}x_2, \ldots, s^{(n)} = s^{(n-1)}x_n$ are all defined and $s^{(n)} = s$. We then say r *is satisfied at s*. Additionally the Todd–Coxeter algorithm ensures that if r' was satisfied at s' before we pushed (r, s) then it will be afterwards. When we have pushed each relator $r \in R$ from each coset $s \in S$ and each word $w \in W$ from s_1 then we will have finished.

To push (r, s), where $r = x_1x_2 \ldots x_n$, we calculate $s^{(0)} = s$, $s^{(i)} = s^{(i-1)}x_i$ until either:

(a) we have found $s^{(n)}$, in which case s and $s^{(n)}$ must be the same coset. We record this as the *Type A coincidence* $s = s^{(n)}$; or
(b) $s^{(j-1)}x_j = 0$ for some $j \leq n$. We then calculate $s'^{(n)} = s$, $s'^{(i-1)} = s'^{(i)}x_i^{-1}$ until either
   (i) we find $s'^{(j-1)}$ when we record the type A coincidence $s^{(j-1)} = s'^{(j-1)}$; or
   (ii) $s'^{(l)}x_l^{-1} = 0$. If $l = j$ we have shown that $s^{(j-1)}x_j$ must be equal to $s'^{(l)}$. We record this as the *Type B coincidence* $s^{(j-1)}x_j = s'^{(l)}$. Otherwise, we define new cosets to be $s^{(j)}, \ldots, s^{(l-1)}$ and record Type B coincidences $s^{(j-1)}x_j = s^{(j)}, \ldots, s^{(l-1)}x_l = s'^{(l)}$.

What this has achieved is to store on the Type A and B coincidence stacks, the extra information which must be added to the coset table to ensure that r will be satisfied at s. A Type A coincidence is what is traditionally meant by a coincidence, while a Type B coincidence is roughly the same as what was previously called a deduction [in Leech (1984) for instance]. We now have to apply this information to the coset table, taking care to preserve the consistency condition and all results "r' is satisfied at s'".

2.4. PROCESSING COINCIDENCES

Type A Coincidence—$s = s'$

(1) If s or s' has been deleted replace them by their undeleted images.
(2) If now $s = s'$ stop, otherwise without loss of generality $s < s'$.
(3) *Empty* s', by generating, for each non-zero entry $f(s', x) = s''$, the Type B coincidence $s'x = s''$ and then setting $f(s', x) = f(s'', x^{-1}) = 0$.

   Note. A little time could be saved by recording $sx = s''$ instead of $s'x = s''$, however, it is convenient in double coset enumeration to have a self-contained routine to empty a row onto the Type B coincidence stack.
(4) Delete the coset s' and record that its undeleted image is s.

Type B coincidence—$sx = s'$

(1) Replace s and s' by their undeleted images.
(2) If $f(s, x) = 0$ (when by the Consistency Condition we also have $f(s', x^{-1}) = 0$), set $f(s, x) = s'$ and $f(s', x^{-1}) = s$ and stop.
(3) If $f(s, x) = s'$ then stop.
(4) Let $s'' = f(s, x)$ generate the Type A coincidence $s' = s''$.

We have thus seen how Type A and B coincidences can each induce the other, so that a single push can induce a large number of coincidences, and substantially reduce the size of S.

2.5. IMPROVEMENTS ON THIS ALGORITHM

The "raw" algorithm described above can be dramatically improved in time and space performance by a number of refinements, all of which apply equally to Double Coset Enumeration.

Lookahead—in many cases, the "raw" algorithm defines a large number of cosets which are not used in discovering the "critical" coincidence which causes the table to collapse to its final form. This problem can be addressed by the programme periodically entering *Lookahead Mode*. In this mode relators are pushed as normal (*Define Mode*), except that if new cosets would have had to be defined, the push is abandoned, generating no coincidence. There are a number of strategies for deciding when to enter and leave lookahead mode.

Weights—This is a novel technique due to Parker for deciding the order in which coset-relator pairs are pushed. Each relator r is assigned a weight w_r and each coset has a weight u_s. Coset-relator pairs are pushed in order of increasing $u_s + w_r$, with the new cosets s' defined during the push having $u_{s'} = u_s + w_r$. With carefully chosen weights w_r this can give a spectacular improvement in performance.

The problem of choosing weights w_r for a presentation, is a tricky one. The time and space required by the algorithm can depend critically on choice of weights. In one example, changing a single weight from 30 to 32 produced a factor of three improvement in performance. It seems reasonable that, in general, longer relations should have higher weights, as they take more resources to push, and are likely to define more cosets; however, beyond that correct weighting strategy is far from obvious.

A technique which seems quite effective, where it can be applied, is to experiment with smaller groups, related to the group in question, in an effort to find "correct" weights there. In the Fischer groups, for example, $2^2.\mathrm{Fi}_{22} = \langle Y_{332} \mid S = 1\rangle$, $2 \times \mathrm{Fi}_{23} = \langle Y_{532} \mid S = 1\rangle$ and $3\mathrm{Fi}_{24} = \langle Y_{552} \mid S = 1\rangle$ [see ATLAS, p. 232, Conway *et al.* (1985) and Conway & Pritchard (1986) for details of this notation]. Appropriate weights for $S = 1$, and for various redundant relators that were used could be found in $2^2.\mathrm{Fi}_{22}$, refined in $2 \times \mathrm{Fi}_{23}$ and used to get reasonable performance on $3\mathrm{Fi}_{24}$.

It has been proposed by Soicher (private communication) that the weights w_r, and possibly also u_s, should be adaptive, so that if a relator produced a large number of coincidences its weight would decrease, and if it caused a large number of cosets to be defined its weight should be increased. This has yet to be implemented.

Early-Closing—This term was coined by Conway, following the observation that in hand-enumerations, the final double coset table was often reached long before the theoretical completion of the enumeration.

We call a coset table *closed* if it has no zeroes in it (so that it represents a permutation action of the free group on X). A closed table will remain closed after future pushes, and can only get smaller. If a lower bound on the number of cosets is known, the enumeration can often be shown to be complete at this point. Alternatively, permutations can be extracted and the relations in R tested using fast permutation-multiplying programs.

Random Strategies—This is another suggestion due to Conway, which has still to be implemented. He suggested that, rather than using weights, or the more conventional approaches to deciding what order to push relators and cosets, they should be pushed in a (suitably biased) random order, until Early-Closing occurs. This strategy is, in fact, particularly appealing for double coset enumeration, since it would avoid a considerable amount of overhead (see section 3.5.).

3. Double Coset Enumeration

3.1. INTRODUCTION

We now consider how our understanding of K can be exploited to save space and time. The group G is now considered to be generated by $K \cup X$ and $X \cap K$ is taken to be empty. For each $x \in X$ we set $L_x = K \cap K^{x^{-1}}$. We then have that $L_x \cong L_{x^{-1}}$ with an isomorphism given by

$$\theta_x \colon L_x \to L_{x^{-1}} l \mapsto l^x$$

We will assume a detailed understanding of the internal structure of K, and of the maps θ_x. We call L_x *the gain group of* x, since the space saving in the description of the action of x on S is roughly proportional to $|L_x|$. For any $s \in S$ and any $l \in L_x$, we have $sx = s' \Rightarrow slx = s'l^x$. We thus need only store the action of x on a set of (H, L_x) double coset representatives. Equivalently, we store the action of x on a set of L_x-orbit representatives in the action on S.

In our example we have

$$X = \{a, b_2, c_2, d_2, e_2, b_3, c_3\} \quad \text{and} \quad L_a = \langle c_1, d_1, e_1 \rangle \cong S_4, L_{b_2} = L_{c_2} = \ldots = L_{c_3} = K.$$

Each θ_x is the identity. This is typical for a Coxeter presentation; these are very "good" presentations for double coset enumeration, since it is usually possible to find a subgroup K, such that most of K commutes with most of X.

3.2. THE DOUBLE COSET TABLE

For each (H, K) double coset $D \subseteq G$, we pick a representative single coset d (d will always be the representative for D, d' for D', etc.). We call the set of double cosets $\mathcal{D}$. Our single cosets s are of the form dk, but these names are not in general unique. Our double coset table has, for each $x \in X$, a set of columns corresponding to a (particular) set of coset representatives for L_x, the rows of the table correspond to $\mathcal{D}$ and the entries are in $(\mathcal{D} \times K) \cup \{0\}$. If the entry in the D row of the kx column (where k is one of the coset representatives for L_x) is $(D', k') \neq 0$, then we read this as indicating that $dkx = d'k'$, otherwise that dkx is unknown.

In our example we thus have $1+1+1+1+1+1+5 = 11$ columns which, coincidentally, is the same as the number needed in single coset enumeration (there are eleven generators of which four are in K). However, there are about 6000 double cosets, compared to at least 300 000 single cosets, giving a roughly 50-fold saving in space.

There is a space saving even when $L_x = \{1\}$ $\forall$ $x \in X$, because no space is needed to store the action of K on S. For example, in the case of a 2,3-group, $G = \langle x, y \mid x^2 = y^3 = 1, \ldots \rangle$, we can take $K = \langle y \rangle$ and have columns for x, yx and y^2x, instead of x, y and y^2, while still reducing the number of rows by a factor of almost three.

In addition we store for each row D the muddle group $M_d \leq K$ given by $M_d = \{k \in K : dk = d\}$. The number of single cosets in D is given by $|K : M_d|$. There are two consistency conditions on the double coset table.

(1) For every entry $dkx = d'k'$, when we work out $d'k'x^{-1}$ we get dk.
(2) For every entry $dkx = d'k'$ $(M_d^k \cap L_x)^x \leq M_{d'}^{k'}$.

The first is the same condition we had on a single coset table, the second is required to ensure that the action of X on the single cosets is well-defined (see section 3.3 below).

We retain the concept of a coset (now a double coset D) being deleted; however, we need to record which single coset $d'k'$ is the same as the representative single coset d of D. The undeleted image of a single coset dk is then $d'(k'k)$ if D′ has not been deleted or $d''(k''k'k)$ if D′ has been deleted and the undeleted image of d' is $d''k''$.

3.3. Pushing Relators

The basic operation in pushing an (r, s)-pair (now an (r, dk)-pair) is that of calculating sx, $s \in S$, $x \in X$ (now dkx, $D \in \mathscr{D}$, $k \in K$, $x \in X$). To do this we find the L_x-coset representative for k, say k' and look in the (x, k') column in the D row. This will tell us that $dk'x = d''k''$, say; but now $k = k'l$, $l \in L_x$, so we can write $dkx = dk'lx = dk'xl^x = d''k''l^x$, and we can compute $k''l^x$ inside K.

When $M_d > \{1\}$, a single coset $dk \subseteq D$ will have $|M_d|$ distinct "names"—$\{dmk : m \in M_d\}$, and we must ensure that computing $dmkx$ yields the same single coset as computing dkx (though possibly with a different name).

Without loss of generality assume that there is an entry $dkx = d'k'$ in the double coset table. We need to ensure that calculating $dmkx$ yields the same single coset $d'k'$ for all $m \in M_d$. We divide cases according to whether $mk \in kL_x$ or not.

In the first case, we compute

$$dmkx = dkm^kx = dkxm^{kx} = d'k'm^{kx},$$

since $mk \in kL_x$, we have $m^k \in M_d^k \cap L_x$ and so, by consistency condition 2, $m^{kx} \in M_{d'}^{k'}$ and $k'm^{kx} \in M_{d'}k'$ and $d'k'm^{kx}$ is the same single coset as $d'k'$.

In the second case, we might be looking up the same single coset image sx using different entries in the double coset table for different names. We avoid this by renaming single cosets so that they are in the earliest possible L_x-coset (in the ordering of the columns of the table).

We define $f_x(k)$ to be the position (in this order) of the coset kL_x. We then define a function

$$\kappa : X \times \{M : M \leq K\} \times K \to K,$$

where $\kappa(x, M, k)$ is the element m of M which minimises $f_x(mk)$. Since left and right regular actions commute, $f_x(mkl) = f_x(mk)$ $\forall$ $l \in L_x$ so $\kappa(x, M, k)$ depends on $f_x(k)$ and not on k. In fact κ (x, M, k) depends only on the column of the double coset table corresponding to k and x and on M. This provides a convenient way of storing and accessing the table in implementation.

What this does is to provide a way of finding a canonical (M_d, L_x)-double coset representative of k. We can then look up dkx in the column corresponding to this, and be sure to get the same result whichever name we choose for our single coset dk. This process is roughly equivalent to Conway's use of β-entries (Conway, 1984) to denote that an entry had been replaced by another, but takes advantage of the full information about the muddle group M_d. The process of calculating dkx is then as follows:

(1) find $f_x(k)$;
(2) lookup $\kappa(x, \mathrm{M}_d, k)$ using $f_x(k)$, getting k' say. Then $k' \in \mathrm{M}_d$ so $dkx = dk'kx$;
(3) find $f_x(k'k) = i$, say;
(4) lookup dk_0x in the double coset table, where k_0 is the ith L_x-coset representative. This is $d''k''$, say;
(5) set $l = k_0^{-1}k'k$. l will be in L_x. So $dkx = dk_0lx = dk_0xl^x$;
(6) then $dkx = d''(k''l^x)$.

3.4. Processing Coincidences

The above enables us to push relators just as for single coset enumeration, obtaining either Type A coincidences—$dk = d'$ or Type B coincidences—$dkx = d'k'$. Type C coincidences also exist. We process these as follows.

Type A—$dk = d'$

(1) Replace dk and d' by their undeleted images.

   Note. This process will yield images $\bar{d}\bar{k}$ and $\bar{d}'\bar{k}'$. We then replace d with $\bar{d}$, k with $\bar{k}\bar{k}'^{-1}$ and d' with $\bar{d}'$.
(2) If $\mathrm{D} = \mathrm{D}'$ generate the *Type C coincidence* $\langle k \rangle$ fixes d. The general form of a Type C coincidence is M fixes d, where $\mathrm{M} \leq \mathrm{K}$ and $\mathrm{D} \in \mathscr{D}$.
(3) Otherwise wlog $\mathrm{D}' > \mathrm{D}$ (inverting k if necessary).
(4) Empty row D′ (as for single cosets, generating Type B coincidences).
(5) Generate the Type C coincidence $\mathrm{M}_{d'}^{k^{-1}}$ fixes d.
(6) Mark D′ as a deleted double coset, with d' replaced by dk.

Type B—$dkx = d'k'$

(1) Replace dk and $d'k'$ by their undeleted images.
(2) If consistency condition (2) is satisfied for the entries corresponding to dkx and $d'k'x^{-1}$ then we can proceed, otherwise we generate the Type C coincidences required—($(\mathrm{M}_d^k \cap \mathrm{L}_x)^{xk'^{-1}}$ fixes d' and/or $(\mathrm{M}_{d'}^{k'} \cap \mathrm{L}_{x^{-1}})^{x^{-1}k^{-1}}$ fixes d), (re)generate the Type B coincidence $dkx = d'k'$ and stop. There is a risk of looping here, which must be avoided. This can be simply done by handling all pending Type C coincidences before any pending Type Bs.
(3) We work out which entries in the double coset table we would use to calculate dkx and $d'k'x^{-1}$ and divide cases.

   (i) Empty entries in distinct places can be filled at once.
   (ii) If at least one entry is non-empty, say $dkx = d''k''$ already, then stack Type A coincidence $d'k'k''^{-1} = d''$.
   (iii) If the entries are empty and in the same place, i.e. $\mathrm{D} = \mathrm{D}'$, $x = x^{-1}$, k and k' are in the same $(\mathrm{M}_d, \mathrm{L}_x)$ double coset, then we can fill in the entry, and we have

additional information. We set k_0 to be the (M_d, L_x) double coset representative for k and k' and suppose $k = mk_0l$ and $k' = m'k_0l'$. Then

$$dk_0lx = dk_0l' \quad \text{and} \quad dk_0l'x = dk_0l,$$

so

$$dk_0x = dk_0l'(l^x)^{-1} = dk_0l(l'^x)^{-1}$$

and

$$dk_0l'(l^x)^{-1}(l'^x)l^{-1}k_0^{-1} = d.$$

We accordingly generate a Type C Coincidence—$\langle k_0l'(l^x)^{-1}(l'^x)l^{-1}k_0^{-1}\rangle$ fixes d.

Type C—M fixes d.

(1) If d has undeleted image $d'k'$, we replace d by d' and M by $M^{k'^{-1}}$.
(2) If $M \le M_d$ then stop.
(3) Empty row D onto the Type B stack.
(4) Set the new value of M_d to $\langle M, M_d\rangle$.

Apart from the risk of looping, the pending coincidences may be cleared in any order. CBA and CAB have been tried, producing similar performance.

3.5. FURTHER REFINEMENTS

The algorithm presented so far is essentially the corrected version of the algorithm proposed by Conway (1984). This saves space when compared with the equivalent single coset enumeration, but takes roughly the same amount of time, since each relator is still pushed from each single coset. When every letter of the word in $K \cup X$, which makes up a relator, commutes with all or part of K, then some of the pushes are redundant and time can be saved.

For each relator r we compute a group $N_r \le K$ as follows.

If $r = w_1w_2 \dots w_l$, where $w_i \in K \cup X$ then we set $N^{(0)} = K$ and for $i = 1, \dots, l$

if $w_i \in K$ then $N^{(i)} = N^{(i-1)w_i}$
if $w_i \in X$ then $N^{(i)} = (N^{(i-1)} \cap L_{w_i})^{w_i}$.

We then set $N_r = N^{(l)}$ and set $N_0 = N_r$ and $N_i = N_{i-1}^{w_i}$ (which we know will be defined and contained in K).

We can see from the construction that for all $n \in N_r$, $n^{w_1 \cdots w_k} \in N_k$ and whenever $w_{k+1} \in X$ we have $N_k \le L_{w_{k+1}}$.

The point of all this is that when we push r from a single coset s and from sn for $n \in N_r$, we have $snw_1 \dots w_k = sw_1 \dots w_kn^{w_1 \cdots w_k}$, so that we look up $snw_1 \dots w_{k+1}$ in the same entry of the double coset table as $sw_1 \dots w_{k+1}$.

So, if r is satisfied at s, it will be satisfied at sn automatically. In general, we have to push each relator $r \in R$, in each double coset $D \in \mathscr{D}$, from dk, where k ranges through a set of (M_d, N_r)-double coset representatives.

This substantially speeds up the algorithm, for presentations where $N_r \ne \{1\}$ for a reasonable number of r, but adds considerable complexity to it. In one example, with $K = S_4$, this technique avoided 5 632 854 redundant pushes, leaving only 1 140 493 that actually had to be done. As remarked in section 2.4, this calculation is not needed if Conway's random strategy is followed.

4. Implementation Considerations

4.1. WORKING IN K

Throughout the description of the Double Coset Enumeration algorithm, we have assumed the ability to work freely with elements and subgroups of K. The implementation of the algorithm by the author used look-up tables, for speed and simplicity of programming. We number the elements of K, with 1 being numbered 1, and we number the subgroups of K (not conjugacy classes, the complete lattice of subgroups) with $\{1\}$ being numbered 1 and K having the largest number. We store the following look-up tables.

The multiplication table of K.
For each $k \in K$ the cyclic subgroup $\langle k \rangle$.
For each $M_1, M_2 \leq K$ the subgroup $M_1 \cap M_2$.
For each $M_1, M_2 \leq K$ the subgroup $\langle M_1, M_2 \rangle$.
For each $k \in K$ the element k^{-1}.
For each $k \in K$ and $M \leq K$ the subgroup M^k.
A list of coset representatives for each $M \leq K$.
A list of elements of each $M \leq K$.

The space required by these tables is the main restriction on the groups K which can be used. For the symmetric group S_5 (120 elements, 156 subgroups), about $\frac{1}{2}$MB of storage space is needed. Tables have been prepared by computer for S_4, A_5 and S_5, and by hand for the smaller groups 2^2, 3, D_8 and S_3.

As well as these tables describing the structure of K as an abstract group we need to describe the action of each $x \in X$ on K as far as it can be seen, and to describe the structure of the double coset table, which depends on this. We number the columns of the double coset table (each column corresponds to an $x \in X$ and one of the coset representatives of L_x in K) and the elements of X. We use the following tables.

For each $x \in X$ the subgroup L_x.
For each $k \in K$ and $x \in X$ the column of the double coset table in which to look up dkx, assuming that $M_d = \{1\}$.
The table for κ—as remarked this has values depending on a column of the double coset table and a subgroup of K.
Tables giving the elements of X and K corresponding to a given column of the double coset table.
For each $x \in X$ and $k \in L_x$ the element k^x.
For each $x \in X$ and $M \leq L_x$ the subgroup M^x.
For each $x \in X$ the generator x^{-1}.

As remarked above, these tables make the algorithm relatively simple to implement, and fast to operate, but limit the size of K.

The technique of weights is implemented by keeping a record of the first (lowest numbered) row D_w with each weight w and of the last (highest numbered) row D_r from which each relator r has been pushed. There is a current weight w_c and each relator r is pushed in turn from the rows $D \in \mathscr{D}$ such that $D_r < D < D_{w_c - w_r + 1}$. Any new cosets defined are placed at the end of the table. When this has been done for each r, w_c is incremented and D_{w_c} is set equal to the number of the first unused row, so that new cosets defined will have weight w_c. This technique will cope unchanged if relator weights, w_r are allowed to vary during an enumeration, but will not allow easily for changes in coset weights, w_c.

4.2. OPTIMISATION TECHNIQUES

Amongst the most complex and time-consuming aspects of the algorithm are the two double coset reductions: to find the correct column, when looking up dkx, and to work out which single cosets from a double coset to push. Both of these processes are quite simple, except when two or more of M_d, L_x and N_r are *proper* subgroups of K. These conditions are quite rare, as in a typical run $L_x = K$ for all but a few x, $M_d = 1$ for most double cosets D, during most of the enumeration and $N_r \in \{\{1\}, K\}$ for most relators. A significant saving in time can be made by checking these conditions, and using simplified code where possible.

In a typical run, 334 964 double coset-relator pairs were handled, of which only 63 433 had both N_r and M_d proper subgroups of K; in all the other cases the simplified code could be used.

Another useful observation is that the coincidence stacks are accessed in a very predictable manner (Last In, First Out) so that they can be efficiently overlayed onto (slower) secondary storage, which frees more main store for the coset table.

5. Results

Results are given for the following presentations [for the notation used see ATLAS, p. 232, or Conway *et al.* (1985*b*), Conway & Pritchard (1986) and Soicher (1987)].

$$2^2.{}^2E_6(2) = \langle Q_{422} \mid V = 1, f_1 = (f_2 f_3 a_1 b_1 c_1 d_1)^5 \rangle,$$
$2 \times Co_1$—see ATLAS, p. 183 and Soicher (1987),
$$Fi_{24} = \langle Y_{542} \mid S = 1, f_1 = f_{12}, (ab_1 c_1 d_1 e_1 f_1 ab_2 c_2 d_2 e_2 ab_3 c_3)^{17} = 1 \rangle,$$
$$Fi_{23} = \langle Y_{532} \mid S = 1, f_1 = f_{12}, f_{21} = 1 \rangle,$$
$$2^6.L_2(7) = \langle x, y \mid x^2 = y^3 = (xy)^7 = [x, y]^8 = 1 \rangle.$$

Group	H	K	Columns	Single cosets	Double cosets	Time taken
$2^2.{}^2E_6(2)$	$2^3.2^{20}.U_6(2)$	S_5	15	3 968 055	39 681	20 min
$2 \times Co_1$	$3.Suz{:}2$	S_5	9	3 091 200	34 213	6 min
Fi_{24}	$2 \times Fi_{23}$	S_5	12	306 936	6 332	12 min
Fi_{23}	$2Fi_{22}$	S_4	11	31 671	2 417	90 s
$2^6.L_2(7)$	1	3	3	10 572	3 584	16 s

(CPU times are measured on one processor of an IBM 3084Q.)

A highly optimised single coset enumeration program [due to Soicher (private communication)] took 8 s to enumerate the 10 572 cosets in the same presentation of $2^6.L_2(7)$ and 167 s to enumerate the 31 671 cosets in the presentation of Fi_{23}.

It can be seen that there is considerable variation in speed, average muddle-group size and other measures. As for single coset enumeration, performance varies wildly between groups of similar size or even between presentations of the same group.

6. Possible Future Improvements

A number of techniques have been proposed to reduce the space used by the look-up tables: by using a representation of elements of K as permutations or matrices, by using the techniques of the SOGOS system (Laue *et al.*, 1984) to work with elements and subgroups of a suitable soluble group, or by using the more general package, such as Cayley (Cannon, 1984), to work with a variety of groups. In evaluating such a proposal we consider the essential operations which must be performed quickly while the enumerator is running:

(a) Operations required in the calculation of dkx:

1. finding the (M_d, L_x)-double coset representative k_0 for $k \in K$ and an element $l \in L_x$ such that $k = mk_0 l$;
2. multiplying two elements of K;
3. conjugating an element of L_x by x;
4. inverting an element of K.

(b) Operations required to see that a coincidence is trivial (most are). The operations listed above plus:

1. checking whether a given element $k \in K$ is in a subgroup $M \leq K$;
2. conjugating a subgroup $M \leq K$ by an element $k \in K$. (In the case where a coset has been deleted.)

(c) Operations required in processing coincidences. All the above plus:

1. finding the cyclic subgroup generated by an element of K;
2. inverting an element of X;
3. intersecting a subgroup $M \leq K$ with a gain group L_x;
4. conjugating a subgroup $M \leq L_x$ by $x \in X$;
5. checking to see if one subgroup contains another;
6. finding the subgroup generated by two subgroups.

(d) Operations required for every double coset pushed. All the above plus:

1. obtain a list of (M_d, N_r)-double coset representatives (not needed by the random version of the algorithm).

(e) Other operations used:

1. to check for early closing a method of calculating the number of entries on the D row which are in use (rather than being masked out by κ) given M_d;
2. once early closing has been found, it is useful to know the index of each M_d in K, so that the number of single cosets can be calculated.

Each proposed representation of K should enable these operations to be performed, but in no case is the approach entirely simple.

Smaller gains may be obtainable by implementing adaptive weights, or by Conway's random strategy. Another possible area for improvement, which has not been investigated in detail, is the order of coincidence processing. Both the order in which the stacks should be cleared and the question of whether Last In First Out stacks are the best approach at all remain open questions.

7. Conclusion

The present program represents a substantial improvement over existing (single) coset enumeration techniques for a rather restricted set of suitable presentations. Gains of as much as 100-fold in space requirements and 10-fold in time requirements are available. There are a number of possibilities for further substantial improvements, mainly in the representation of the subgroup K, which could lead to much larger gains.

References

Cannon, J. J. (1984). An introduction to the group theory language Cayley. In: *Computational Group Theory* (M. Atkinson, ed.), pp. 145–183. London: Academic Press.

Cannon, J. J., Dimino, L. A., Havas, G., Watson, J. M. (1973). Implementation and analysis of the Todd–Coxeter algorithm. *Math. Comp.* **27**, 463–490.

Conway, J. H. (1984). An algorithm for double coset enumeration? In: *Computational Group Theory* (M. Atkinson, ed.), pp. 33–37. London: Academic Press.

Conway, J. H., Pritchard, A. D. (1986). Hyperbolic reflections for the Bimonster and $3Fi_{24}$, preprint. *J. Algebra*, to be published.

Conway, J. H., Curtis, R. T., Norton, S. P., Parker, R. A., Wilson, R. A. (1985*a*). *An ATLAS of Finite Groups.* Oxford: Oxford University Press.

Conway, J. H., Norton, S. P., Soicher, L. H. (1985*b*). The Bimonster, the group Y_{555}, and the projective plane of order 3. *Proc. "Computers in Algebra"*, Chicago.

Laue, R., Neubüser, J., Schoenwaelder, U. (1984). Algorithms for finite soluble groups and the SOGOS system. In: *Computational Group Theory* (M. Atkinson, ed.), pp. 105–135. London: Academic Press.

Leech, J. (1963). Coset enumeration on digital computers. *Proc. Camb. Phil. Soc.* **59**, 257–267.

Leech, J. (1984). Coset enumeration. In: *Computational Group Theory* (M. Atkinson, ed.), pp. 3–18. London: Academic Press.

Rotman, J. J. (1984). *An Introduction to the Theory of Groups.* Reading, MA: Allyn & Bacon.

Soicher, L. H. (1987). Presentations for Conway's group Co_1. *Proc. Camb. Phil. Soc.* **102**, 1–3.