GENERALISED SIFTING IN BLACK-BOX GROUPS

SOPHIE AMBROSE, MAX NEUNHÖFFER, CHERYL E. PRAEGER AND
CSABA SCHNEIDER

Abstract

This paper presents a generalisation of the sifting procedure introduced originally by Sims for computation with finite permutation groups, and now used for many computational procedures for groups, such as membership testing and finding group orders. The new procedure is a Monte Carlo algorithm, and it is presented and analysed in the context of black-box groups. It is based on a chain of subsets instead of a subgroup chain. Two general versions of the procedure are worked out in detail, and applications are given for membership tests for several of the sporadic simple groups. The authors' major objective was that the procedures could be proved to be Monte Carlo algorithms, and the costs computed. In addition, they explicitly determined suitable subset chains for six of the sporadic groups, and then implemented the algorithms involving these chains in the GAP computational algebra system. It turns out that sample implementations perform well in practice. The implementations will be made available publicly in the form of a GAP package.

1. *Introduction*

We generalise a sifting procedure, originally introduced by Sims [15, Section 4] (see also [16, Section 2] and [14, Chapter 4]) for computation with permutation groups. Our version is given in the context of black-box groups, and is based on a chain of subsets rather than a subgroup chain. The essential ingredient is a scheme for sifting a group element g down a descending chain

$$G_0 = S_0 \supset S_1 \supset \cdots \supset S_k \qquad (1)$$

of non-empty subsets of a subgroup G_0 of a finite group G. The sifting procedure seeks elements $s_0, \ldots, s_k \in G_0$ such that, for each $i < k$, $S_i s_i \subseteq S_i$ and $g s_0 \ldots s_i \in S_{i+1}$; in addition $g s_0 \cdots s_{k-1} s_k = 1$, and s_k or its inverse lies in S_k. In many instances the s_i will lie in S_i, but this is not required in general. (Conditions on membership for the s_i are given in Definition 4.1(c).)

A major objective of this work is to give a careful presentation of a randomised generalised sifting algorithm with an analysis that proves a guaranteed upper bound on the probability of failure, and provides an estimate of the complexity in terms of the input size. We present our results in a sequence of steps. This 'modular' approach enables us to focus in our exposition on the new concepts and methods introduced at each stage. First we present in Section 4 a skeleton version of the generalised sifting algorithm SIFT that involves a sequence of basic modules, namely various versions of a procedure called BASICSIFT, for which only the input

and output requirements are given explicitly. We prove in Theorem 4.2 that the algorithm SIFT is a Las Vegas algorithm.

Next, in Sections 5 and 6, we present more details of the versions of BASICSIFT that we have developed, and we prove in Theorems 5.3 and 5.6 that for these versions, BASICSIFT is a Monte Carlo algorithm. This exposition of BASICSIFT is given in terms of a generic membership test ISMEMBER, for which only the input and output requirements are given explicitly. Note that the BASICSIFT modules will often be Monte Carlo algorithms with a non-zero probability of returning an incorrect result. However, the complete algorithm SIFT is a Las Vegas algorithm since we can test with certainty that, for our output element $x = s_0 \ldots s_k$, the element gx is equal to the identity. (See Definition 3.4 for a discussion of these types of algorithms.)

In Section 7, we introduce a version of ISMEMBER based on random conjugates. It was this version that inspired the development of the conceptual framework presented in the paper. The idea can best be understood by briefly considering the following special case. Suppose that a finite group G has a chain of subgroups

$$G = H_0 > H_1 > \ldots > H_k = \{1_G\}, \tag{2}$$

and that $a \in H_{k-1} \setminus \{1\}$ is such that, for each i, the subset $a^G \cap H_i$ of a-conjugates lying in H_i forms a single H_i-conjugacy class a^{H_i}. Then, for $x \in G$, the conjugate a^x lies in H_i if and only if $a^x = a^h$ for some $h \in H_i$, and, in turn, this holds if and only if $xh^{-1} \in C_G(a)$. Thus $a^x \in H_i$ if and only if $x \in C_G(a)H_i$; that is to say, a membership test for a^x to lie in the subgroup H_i is equivalent to a membership test for x to lie in the subset $C_G(a)H_i$. The development of this idea to handle the general case where the subsets $a^G \cap H_i$ split into several H_i-conjugacy classes led to the theory presented in Section 7.

In Section 8, we give full details of a version of ISMEMBER that relies on element orders. For the corresponding version of BASICSIFT we are then able to provide, in Corollary 8.2, our most comprehensive complexity estimate.

Before presenting the theoretical details, we give a worked example of our algorithm for the Higman–Sims sporadic simple group in Section 2. This example was chosen to illustrate most of the methods that will be developed in the paper.

The original motivation for this research stemmed from the matrix group recognition project (see [11, 12]), and in particular from the need to recognize constructively all quasi-simple matrix groups over finite fields. The usual approach has been to design algorithms for recognizing finite quasi-simple groups by their intrinsic properties as abstract groups, rather than building different algorithms for each of their different matrix representations. This has resulted in the development of recognition algorithms for most of the almost simple groups represented as black-box groups (see [1, 2, 4, 5, 6, 10]). A black-box group is one in which the elements are represented (possibly non-uniquely) as binary strings of bounded length, and in which we can perform the following operations (and only these): we can test whether two given strings represent the same group element, and we can produce strings representing, respectively, the inverse of a given element, and the product of two given elements. In this paper we give algorithms that involve only these 'black-box operations' of equality tests, extracting inverses, and multiplying group elements. Thus our algorithms are black-box algorithms.

We are aware of the impressively successful practical algorithms, as yet unpublished, of P. E. Holmes, S. A. Linton, E. A. O'Brien, A. J. E. Ryba and R. A. Wilson [9] for recognizing sporadic groups based on the theory of involution centralisers. However, there seemed to be no framework available to analyse the probability of completion or the cost of these

algorithms. Our motivation was based on both experience and hope: experience with developing recognition algorithms for finite symmetric and alternating groups in [**1, 2**], complete with proofs and complexity analyses; and hope that the ideas of Charles Sims could be made effective for black-box groups, where the information needed about a permutation or matrix action must be derived from purely group-theoretic properties. Success in computing with some of the sporadic simple groups suggested that our new approach would provide an alternative method for recognizing and computing with these groups. We believe that we have been successful, both theoretically and in practice. The algorithmic framework presented in this paper offers an effective and convenient means of analysing membership tests for sporadic simple groups and other groups, providing proofs of completion probability and complexity. The framework offers flexibility in choice of subset chains and types of the basic sifting procedures. Explicit examples of the algorithms have been developed and implemented for several of the sporadic groups, and perform very well in practice. In Section 10 we summarise the information about these examples, and we also present some details concerning the implementations of the procedures presented in this paper; illustrative tables are included in Appendix A. We emphasise that all the groups that occur in this paper are finite.

2. *Generalised sifting: an example*

The aim of this section is to explain our approach using the example of the Higman–Sims group HS. We think of HS as a group given to us in its most natural representation: that is, a group of permutations with degree 100. Throughout this section we use various facts concerning HS, and the validity of these facts can easily be checked using the ATLAS [**7**], or a computer algebra package, such as GAP [**8**] or MAGMA [**3**]. In order to describe subgroups of HS, we use the notation introduced in the ATLAS.

Suppose that a and b are standard generators in the sense of [**17**] for HS given on the ATLAS web site [**18**]. Assume that G is a black-box group isomorphic to HS, and that x and y are standard generators for G obtained using the procedure described in the online ATLAS [**18**]. Then the map $a \mapsto x$, $b \mapsto y$ can be extended in a unique way to an isomorphism $\varphi : \text{HS} \longrightarrow G$. Since HS is a permutation group, it is possible to compute, using the Schreier–Sims algorithm, a base and a strong generating set for HS. Using these, a permutation in HS can be efficiently written as a word in a and b. Thus, if $u \in \text{HS}$ then $\varphi(u)$, as a word in x and y, can be computed efficiently. The constructive recognition of the black-box group G requires us to perform the opposite process: given $g \in G$, we must find an element $u \in \text{HS}$ such that $\varphi(u) = g$. This is equivalent to writing the element g as a word in x and y.

In order to complete our task, we specify some (precomputed and stored) elements and subgroups in G. We use the following important convention.

> Every element that we introduce in G from now on will be expressed as a word in x, y. Similarly, every subgroup of G that we use will be given with a generating set, and each generator in this set is assumed to be a word in x, y.

Let L_1 be a maximal subgroup of G isomorphic to $U_3(5).2$. A generating set for such a subgroup can be found by computing a generating set for a maximal subgroup in HS isomorphic to $U_3(5).2$, and mapping the generators into G using φ. In the same way, we find a maximal subgroup L_2 in L_1 isomorphic to $5^{1+2} : (8 : 2)$. Let L_3 be a cyclic subgroup of L_2 of order 8 in a complement $8 : 2$ for 5^{1+2}. To be consistent with the notation to be introduced in later sections of the paper, we will denote a generator of L_3 by a. We emphasise

that this element a lies in L_3, and is not a standard generator of HS. Let a be an element of order 8 in L_3, and set $L_4 = 1$. The four generators of L_3 are all conjugate to each other in G and in L_1; they fall into two conjugacy classes of L_2; and they are pairwise not conjugate in L_3. Thus there are elements $t_1 \in L_1$ and $t_2, t_3, t_4 \in L_2$ such that $a^G \cap L_2 = a^{L_2} \cup a^{t_1 L_2}$, $a^{L_2} \cap L_3 = \{a, a^{t_2}\}$ and $a^{t_1 L_2} \cap L_3 = \{a^{t_1 t_3}, a^{t_1 t_4}\}$. Set $\mathcal{T}_2 = \{1, t_1\}$, $\mathcal{T}_3 = \{1, t_2, t_1 t_3, t_1 t_4\}$, and $\mathcal{T}_4 = \{1\}$.

We therefore have a chain of subgroups

$$G \geqslant L_1 \geqslant L_2 \geqslant L_3 \geqslant L_4 = 1,$$

with $|G : L_1| = 176$, $|L_1 : L_2| = 126$, $|L_2 : L_3| = 250$ and $|L_3| = 8$.

2.1. *Sifting $g \in G$ into the first subset: element orders*

Let $g \in G$. If we were to perform Sims's usual sifting procedure, we would look for an element $h_1 \in G$ such that $gh_1 \in L_1$. The probability that a random h_1 satisfies this property is $|L_1|/|G| = 1/176$. What we do instead is as follows. Let $C_1 = C_G(a)$. We look for an element $h_1 \in G$ such that $gh_1 \in C_1 L_1$. As $|C_1| = 16$ and $|C_1 \cap L_1| = |C_{L_1}(a)| = 8$, the probability that $gh_1 \in C_1 L_1$, for a random h_1, is $|C_1 L_1|/|G| = 2|L_1|/|G| = 1/88$.

In order to make this work, we must have a membership test for $C_1 L_1$. Since $a^G \cap L_1 = a^{L_1}$, we know, as explained in the introduction, that, for $u \in G$, $u \in C_1 L_1$ if and only if $a^u \in L_1$. Thus, to obtain a membership test for $C_1 L_1$, we need only to design a membership test for L_1. Let $u \in G$, and let X_1 be a generating set for L_1; set $\overline{X_1} = X_1 \cup \{u\}$. It is clear that $u \in L_1$ if and only if $\langle \overline{X_1} \rangle = \langle X_1 \rangle$. Now about one quarter of the elements of G have order 15 or 11, but no element in L_1 has order equal to one of these numbers. Hence we select random elements in $\langle \overline{X_1} \rangle$. If such a random element has order 11 or 15, then we conclude with certainty that $u \notin L_1$. If, however, after many random selections we do not find an element with order 11 or 15, then we may say that $u \in L_1$ with a certain high probability. This can be formulated to give a one-sided Monte Carlo membership test for $C_1 L_1$; see Section 8 for details.

2.2. *Sifting gh_1 into the second subset: random conjugates*

The intersection $a^{L_1} \cap L_2$ is the union of two conjugacy classes in L_2, namely a^{L_2} and $a^{t_1 L_2}$, where $t_1 \in L_1$ and we set $\mathcal{T}_2 = \{1, t_1\}$ as above. Let C_2 denote the set $C_G(a)\mathcal{T}_2$. As $L_2 \leqslant L_1$ and $\mathcal{T}_2 \subset L_1$, we have $C_2 L_2 \subset C_1 L_1$. Now we seek an element $h_2 \in L_1$ such that $gh_1 h_2 \in C_2 L_2$. We will call an element $h_2 \in L_1$ 'good' if and only if $gh_1 h_2 \in C_2 L_2$ or, equivalently, if and only if $a^{gh_1 h_2} \in L_2$. If h_2 is a uniformly distributed random element of L_1 and $gh_1 \in C_1 L_1$, then $a^{gh_1 h_2}$ is a uniformly distributed random element of the conjugacy class a^{L_1}. For each $x \in a^{L_1} \cap L_2$ there are $|C_{L_1}(a)|$ choices for $h_2 \in L_1$ such that $a^{gh_1 h_2} = x$. Therefore the total number of 'good' elements is $|a^{L_1} \cap L_2| |C_{L_1}(a)|$, and so the probability that h_2 is 'good' is $|a^{L_1} \cap L_2| |C_{L_1}(a)|/|L_1| = |a^{L_1} \cap L_2|/|a^{L_1}| = 500/31500 = 1/63$.

In order to test whether $a^{gh_1 h_2} \in L_2$, recall that L_2 is isomorphic to $5^{1+2} : (8 : 2)$. A deterministic membership test for L_2 can easily be designed using the fact that $N_G(Z(5^{1+2})) = L_2$ where $Z(5^{1+2}) = \langle b \rangle$ is the centre of 5^{1+2}; in other words, to test whether an element $x \in G$ lies in L_2, simply test whether $b^x \in \{b, b^2, b^3, b^4\}$.

2.3. *Sifting $gh_1 h_2$ into the third subset*

The group L_3 is cyclic with order 8. Set $C_3 = C_G(a)\mathcal{T}_3$, where $\mathcal{T}_3 = \{1, t_2, t_1 t_3, t_1 t_4\}$. As $\mathcal{T}_3 \subset \mathcal{T}_2 L_2$, we obtain $C_3 L_3 \subset C_2 L_2$. We look for an element $h_3 \in L_2$ such that, given

$gh_1h_2 \in C_2L_2$, we have $gh_1h_2h_3 \in C_3L_3$. Using the definition of $\mathcal{T}_3$, we find that, given $gh_1h_2 \in C_2L_2$, the condition $gh_1h_2h_3 \in C_3L_3$ holds if and only if $a^{gh_1h_2h_3} \in L_3$. Arguing as for the previous case, the probability that, given $gh_1h_2 \in C_2L_2$, a random $h_3 \in L_2$ yields $gh_1h_2h_3 \in C_3L_3$ is at least $\min\{|a^{L_2} \cap L_3|/|a^{L_2}|, |a^{t_1L_2} \cap L_3|/|a^{t_1L_2}|\}$. It is easy to compute that this number is $2/250 = 1/125$. At the end of this process we have, with high probability, $a^{gh_1h_2h_3} \in \{a, a^{t_2}, a^{t_1t_3}, a^{t_1t_4}\}$. Therefore, after a number of equality tests we obtain a word w in x, y such that $gw \in C_4$ where $C_4 = C_G(a)$. As $|C_G(a)| = 16$, using the map φ, it is easy to compute each element of $C_G(a)$ as a word in x, y. Then, comparing gw against the elements of $C_G(a)$, it is now easy to express g as a word in x, y.

Thus the main ingredients of this process are a descending chain of subgroups $\{L_i\}_{i=1}^4$, a sequence of subsets $\{C_i\}_{i=1}^4$ defined in terms of the centraliser of the element a, and the sequence $\{\mathcal{T}_i\}_{i=1}^4$ of subsets where we take $\mathcal{T}_1 = \{1\}$. Our sifting procedure progressed through the following descending chain of non-empty subsets:

$$G \supset C_1L_1 \supset C_2L_2 \supset C_3L_3 \supset C_4;$$

the final step is a series of equality tests with the elements of C_4.

3. *A small toolbox*

In this section we collect several results that we need in our proofs. For an event E, $\mathrm{Prob}(E)$ denotes the probability of E. For events A and B, $\mathrm{Prob}(A|B)$ denotes the conditional probability of A, given that B holds. We recall that $\mathrm{Prob}(A|B) = \mathrm{Prob}(A \cap B)/\mathrm{Prob}(B)$. The following result from elementary probability theory will often be used in this paper.

LEMMA 3.1. *If A, B and C are events such that $C \subseteq B \subseteq A$, then*

$$\mathrm{Prob}(C|A) = \mathrm{Prob}(C|B) \cdot \mathrm{Prob}(B|A).$$

Proof. As $B = B \cap A$ and $C = C \cap B = C \cap A$, we obtain

$$\begin{aligned}
\mathrm{Prob}(C|B) \cdot \mathrm{Prob}(B|A) &= \frac{\mathrm{Prob}(C \cap B)}{\mathrm{Prob}(B)} \cdot \frac{\mathrm{Prob}(B \cap A)}{\mathrm{Prob}(A)} \\
&= \frac{\mathrm{Prob}(C \cap A)}{\mathrm{Prob}(B)} \cdot \frac{\mathrm{Prob}(B)}{\mathrm{Prob}(A)} = \frac{\mathrm{Prob}(C \cap A)}{\mathrm{Prob}(A)} \\
&= \mathrm{Prob}(C|A). \qquad \square
\end{aligned}$$

LEMMA 3.2. *If $0 \leqslant x < 1$, then $\log((1 - x)^{-1}) \geqslant x$.*

Proof. Observe that the function $f(x) = x - \log((1 - x)^{-1})$ is strictly decreasing for $0 \leqslant x < 1$ and $f(0) = 0$. $\qquad \square$

The following is a general version of Dedekind's modular law. Its proof can be carried out by following that of [**13**, 1.3.14].

LEMMA 3.3. *If U and V are subsets and Z is a subgroup of a group such that $VZ \subseteq V$, then $(V \cap U)Z = V \cap (UZ)$.*

In this paper we use several types of randomised algorithms (that is, algorithms that involve a random choice at some point, so that they do not behave in the same way every time the algorithm is run). We also use algorithms that involve no random choices (that is, deterministic algorithms). We collect together here the definitions of these types of algorithms. To aid our exposition, we give slightly different definitions of these algorithm types than normal, and we comment on the differences below.

DEFINITION 3.4.   (a) Let ε be a real number satisfying $0 \leqslant \varepsilon < 1/2$. A *Monte Carlo algorithm* with 'error probability' ε is an algorithm that always terminates after a finite number of steps, such that the probability that the algorithm gives an incorrect answer is at most ε.

(b) A *one-sided Monte Carlo* algorithm is a Monte Carlo algorithm that has two types of output (typically 'yes' and 'no'), and one of the answers is guaranteed to be correct.

(c) A *Las Vegas algorithm* with 'failure probability' ε (where $0 \leqslant \varepsilon < 1/2$) terminates after a finite number of steps and either returns an answer, or reports failure. An answer, if given, is always correct, while the probability that the algorithm reports failure is at most ε.

(d) For the purposes of this paper, a *deterministic algorithm* is a Monte Carlo algorithm for which the 'error probability' ε is 0 or, equivalently, a Las Vegas algorithm for which the 'failure probability' ε is 0.

Note that our definitions of Monte Carlo and Las Vegas algorithms vary from the usual ones, in that we allow ε to be zero. The reason for this is that some versions of our BASICSIFT algorithm may be deterministic (that is, have zero probability of failure or of returning an incorrect answer). For ease of exposition we decided to treat such an algorithm as a special case of a Monte Carlo or Las Vegas algorithm.

4. *The generalised sifting algorithm*

In this section we present an algorithm for sifting an element g of a finite group G down a (given and precomputed) descending chain (1) of subsets of a subgroup G_0 of G. The algorithm returns either FAIL, or a word $x = s_0 \ldots s_k \in G_0$ such that $gx = 1$, $S_i s_i \subseteq S_i$ for each $i < k$, and s_k or its inverse lies in S_k. If $g \in G_0$, then (see Theorem 4.2) the probability that the algorithm returns FAIL is proved to be at most some pre-assigned quantity ε. Usually, the s_i are returned as words in a given set Y of generators for G_0, or as straight-line programs from the given generating set Y. The algorithm is applied in one of the following contexts.

(1) The element g is known to lie in G_0, and the purpose of the algorithm is to express g as a word in a given generating set. In this context, Theorem 4.2 proves that the algorithm fails with probability at most ε, for some pre-assigned non-negative real number $\varepsilon < 1/2$. Hence, in this context, Algorithm 1 is a *Las Vegas algorithm*.

(2) We assume only that $g \in G$, and the aim is to discover whether or not g lies in G_0. In this context, Theorem 4.2 proves that if the algorithm returns an expression for g, then g must lie in G_0. On the other hand, if the algorithm returns FAIL, then the element g may or may not lie in G_0. Moreover, if $g \in G_0$, then the probability that the algorithm will return FAIL is less than some pre-assigned real number ε, where $0 \leqslant \varepsilon < 1/2$. Hence, in this context (if we interpret the result FAIL as a finding that $g \notin G_0$), Algorithm 1 is a *one-sided Monte Carlo algorithm*.

In either case, we allow the probability bound ε to be zero, and in this situation the resulting algorithm is deterministic. The basic building block for our algorithm is described in the following definition.

DEFINITION 4.1. A 4-tuple $(G_0, H, K, \text{BASICSIFT})$ is said to satisfy the *basic sift condition* in a group G if the following statements hold.

(a) G is a finite group with a subgroup G_0.

(b) H and K are non-empty subsets of G_0 such that either $K = \{1\}$ or $K \subset H$.

(c) BASICSIFT is a Monte Carlo algorithm whose input is a pair (g, ε), where $g \in G$ and ε is a non-negative real number. It satisfies the following condition, either for all inputs $(g, 0)$

Algorithm 1: SIFT
/*see Theorem 4.2 for notation */
Input: $g \in G$ and $(\varepsilon_0, \ldots, \varepsilon_k)$ with $\varepsilon_i \geqslant 0$ and $\sum_i \varepsilon_i < 1/2$;
Output: either $x = s_0 \cdots s_k$ with $S_i s_i \subseteq S_i$ for $i < k$ and $gx = 1$, or FAIL;
set $x = 1$;
for $i = 0$ **to** k **do**
    set $s_i = \text{BASICSIFT}_i(gx, \varepsilon_i)$;
    if $s_i = $ FAIL **then**
        return FAIL
    else
        set $x = xs_i$
    end
end
if $gx \neq 1$ **then**
    return FAIL
else
    return x
end

Algorithm 1: The generalised sifting algorithm.

(in which case it is a deterministic algorithm), or for all inputs (g, ε) with $0 < \varepsilon < 1/2$: *the output y is either* FAIL, *or an element of G_0 such that $Hy \subseteq H$ (if $K \subset H$) or $y^{-1} \in H$ (if $K = \{1\} \not\subset H$). Moreover, if $g \in H$, then* $\text{Prob}(y = \text{FAIL}, \text{ or } (y \in G_0 \text{ and } gy \notin K)) \leqslant \varepsilon$.

To avoid confusion, we comment on the formulation of the condition in Definition 4.1(c). Note that H is in general not a subgroup, and hence $Hy \subseteq H$, for $y \in G$, does not imply that either of y or y^{-1} lies in H. After considering many special cases, we realised that the set inclusion $Hy \subseteq H$ was the appropriate requirement.

Suppose that G is a finite group with a subgroup G_0, and that

$$G_0 = S_0 \supset S_1 \supset \cdots \supset S_{k-1} \supset S_k$$

is a chain of non-empty subsets of G, and set $S_{k+1} = \{1\}$. Suppose further that, for $i = 0, \ldots, k$, BASICSIFT$_i$ is an algorithm such that $(G_0, S_i, S_{i+1}, \text{BASICSIFT}_i)$ satisfies the basic sift condition in G. Then there is a Las Vegas algorithm that, for a given $g \in G$, returns either 'failure' or an element $s_0 s_1 \ldots s_k$ of G_0 such that $S_i s_i \subseteq S_i$ for each $i < k$, the element s_k or its inverse lies in S_k, and $gs_0 s_1 \cdots s_k = 1$. Indeed, as shown in Theorem 4.2, Algorithm 1 has this property.

THEOREM 4.2. *Suppose that G, G_0, S_0, \ldots, S_{k+1}, and* BASICSIFT$_0$, \ldots, BASICSIFT$_k$ *are as in the previous paragraph, and let* SIFT *denote Algorithm* 1. *Let $g \in G$ and $\varepsilon_0, \ldots, \varepsilon_k$ be non-negative real numbers such that $\sum_i \varepsilon_i < 1/2$. Then the following statements hold.*

(i) *If* SIFT$(g, (\varepsilon_0, \ldots, \varepsilon_k))$ *returns a group element x, then $g = x^{-1} \in G_0$ and $x = s_0 s_1 \cdots s_k$, where $S_i s_i \subseteq S_i$ for each $i \in \{0, 1, \ldots, k-1\}$, and $s_k \in S_k$ if S_k contains 1, while $s_k^{-1} \in S_k$ otherwise.*

(ii) *The conditional probability that* SIFT$(g, (\varepsilon_0, \ldots, \varepsilon_k))$ *returns* FAIL, *given that $g \in G_0$, is at most $\sum_i \varepsilon_i$.*

Proof. (i) Suppose that a group element $x = s_0 s_1 \ldots s_k$ is returned. Then the s_i are group elements computed as in Algorithm 1. From Definition 4.1(c), since each s_i is a group element, we find that $S_i s_i \subseteq S_i$ for each $i \in \{0, 1, \ldots, k-1\}$, and also for $i = k$ if $1 \in S_k$; while if $1 \notin S_k$, then $s_k^{-1} \in S_k$. Further, if $1 \in S_k$, then $S_k s_k$ contains s_k, and hence S_k contains s_k. Finally, for each i, s_i lies in G_0 since the algorithm BASICSIFT$_i$ involves selections from the group G_0. Moreover, by the last **if** statement of Algorithm 1 we have $gx = 1$, so that $g = x^{-1} \in G_0$.

(ii) Let E_0 denote the event that $g \in G_0$, and recall that $G_0 = S_0$ and $S_{k+1} = \{1\}$. For each $i = 1, \ldots, k$, let E_i denote the event that the ith execution of the **for** loop in Algorithm 1 is attempted, is successful and returns a correct answer. In other words,

$$E_i : E_{i-1} \text{ holds}, \quad S_j s_j \subseteq S_j \text{ for all } j = 0, \ldots, i-1, \quad \text{and} \quad gs_0 \ldots s_{i-1} \in S_i.$$

Also define E_{k+1} to be the event that the final execution of the **for** loop is attempted, is successful and returns a correct answer. That is,

$$E_{k+1} : E_k \text{ holds}, \quad S_j s_j \subseteq S_j \text{ for all } j = 0, \ldots, k-1, \quad \text{and} \quad gs_0 \cdots s_k = 1.$$

Then the probability that Algorithm 1 returns $x = s_0 s_1 \ldots s_k$ with $S_i s_i \subseteq S_i$ for all $i = 0, \ldots, k-1$, and $gx = 1$, given that $g \in G_0$, is, by definition, $\mathrm{Prob}(E_{k+1}|E_0)$.

Now $E_{k+1} \subseteq E_k \subseteq \cdots \subseteq E_0$, and hence by several applications of Lemma 3.1, we see that

$$\mathrm{Prob}(E_{k+1}|E_0) = \prod_{i=0}^{k} \mathrm{Prob}(E_{i+1}|E_i).$$

Since $(G_0, S_i, S_{i+1}, \text{BASICSIFT}_i)$ satisfies the basic sift condition in G for each $i = 0, \ldots, k$, we have $\mathrm{Prob}(E_{i+1}|E_i) \geqslant 1 - \varepsilon_i$ for each $i = 0, \ldots, k$. Hence

$$\mathrm{Prob}(E_{k+1}|E_0) \geqslant \prod_{i=0}^{k} (1 - \varepsilon_i).$$

Since $0 \leqslant \varepsilon_i < 1$ for all i, we have $\prod_i (1 - \varepsilon_i) \geqslant 1 - \sum_i \varepsilon_i$ (use induction on k), and hence the required probability in part (ii) is at most $\sum_i \varepsilon_i$. □

Algorithm 1 allows different types of algorithms to be used for different links of the chain. For example, if S_k is small, then BASICSIFT$_k$ relies sometimes on nothing more than an exhaustive search through the elements of S_k with the parameter $\varepsilon_k = 0$. Two special types of BASICSIFT algorithms are described in detail in Sections 7 and 8. We first explore their common properties as one-sided Monte Carlo algorithms in Sections 5 and 6.

5. BASICSIFT: *a general approach*

In this section we present a general approach to designing a 4-tuple that satisfies the basic sift condition. The results of this section will become relevant in the discussion of the two algorithms in Sections 7 and 8. We will use one of the general methods given in this section in nearly all cases when we wish to sift an element of S_i into the next subset S_{i+1} in a subset chain (1). The exceptional case occurs when $1 \notin S_i$ and $S_{i+1} = \{1_G\}$, and, as we mentioned at the end of the previous section, in this exceptional case we would typically use an exhaustive search through S_i to find the required 'sifting element'.

Our general approach assumes that we are able to test membership in each of the S_i, and to select a uniformly distributed random element from some subset 'related to' S_i in the chain (1); see Section 2 for examples.

DEFINITION 5.1. A 4-tuple $(G_0, H, K, \text{IsMember})$ is said to satisfy the *membership test condition in G* if the following statements hold.

(a) G and G_0 are finite groups such that $G_0 \leqslant G$.

(b) H and K are non-empty subsets of G_0 such that $H \supset K$.

(c) IsMember is a one-sided Monte Carlo algorithm whose input is a pair (y, e), where $y \in G$ and e is a non-negative real number. It satisfies the following condition, either for all inputs $(y, 0)$ (in which case it is a deterministic algorithm), or for all inputs (y, e) with $0 < e < 1/2$. The output is either TRUE or FALSE; moreover, if $y \in K$ then the output is TRUE, and also Prob(output is TRUE$|y \in H \setminus K) \leqslant e$.

NOTE. For an enhanced version of an IsMember test giving back additional information for later use, consult the examples for M_{11} and Ly in Section 10.

We show that if a 4-tuple $(G_0, H, K, \text{IsMember})$ satisfies the membership test condition in a group G, then we can design an algorithm BASICSIFT such that $(G_0, H, K, \text{BasicSift})$ satisfies the basic sift condition in G. As mentioned above, we assume that we can select uniformly distributed random elements from some subset L of G 'related to' the subset H. The most general conditions that the subset L must satisfy are given in the following definition.

DEFINITION 5.2. Suppose that G is a finite group and $H, K, L \subseteq G$. We say that (H, K, L) is a *sifting triple* if

$$HL \subseteq H, \quad \text{and} \quad \text{for all } h \in H, \ hL \cap K \neq \varnothing. \tag{3}$$

The reason why we introduce the subset L in a sifting triple is that it is rarely possible to make random selections from arbitrary subsets of G, such as H, but we can often make random selections from subgroups. Thus one choice for L is a subgroup satisfying (3). Moreover, we can sometimes obtain a more-efficient algorithm by restricting to a 'nice subset' L of such a subgroup, provided that we can still make random selections from L. Sometimes this is possible simply because L is small enough to be held in the memory. In the latter case we do not have to perform a random search, but can use an exhaustive search. This is analysed in Section 5.2.

If (H, K, L) is a sifting triple, then the number

$$p(H, K, L) = \min_{h \in H} \frac{|hL \cap K|}{|L|}$$

is called the *sifting parameter*. We note that the definition of a sifting triple implies that $p(H, K, L) > 0$. The sifting parameter plays an important rôle in estimating the complexity of Algorithm 1.

5.1. *A BASICSIFT algorithm using random search*

THEOREM 5.3. *Suppose that* $(G_0, H, K, \text{IsMember})$ *satisfies the membership test condition in a group G, and that L is a subgroup of* G_0 *such that* (H, K, L) *is a sifting triple. If* RANDOMELEMENT(L) *returns uniformly distributed, independent random elements of L, and* BASICSIFT *is Algorithm 2, then the 4-tuple* $(G_0, H, K, \text{BasicSift})$ *satisfies the basic sift condition in G. Moreover, the cost of executing* BASICSIFTRANDOM$(\cdot, \varepsilon)$ *is at most*

$$O\left(\log(\varepsilon^{-1})\, p^{-1}\, (\xi + \varrho + \nu(e))\right),$$

Algorithm 2: BASICSIFTRANDOM
```
/*See Theorem 5.3 for notation */
```

Input: (x, ε) where $x \in G$, and $0 < \varepsilon < 1/2$;
Output: y, where either $y = $ FAIL, or $y \in S$;
set $e = \begin{cases} 0 & \text{if ISMEMBER is deterministic,} \\ \varepsilon p/(2(1-p)) & \text{otherwise;} \end{cases}$
set $N = \begin{cases} \lceil \log(\varepsilon)/\log(1-p) \rceil & \text{if ISMEMBER is deterministic,} \\ \lceil \log(\varepsilon/2)/\log(1-p) \rceil & \text{otherwise;} \end{cases}$
set $n = 0$;
repeat
    set $y = $ RANDOMELEMENT(L);
    if ISMEMBER $(xy, e) = $ TRUE **then**
        return y
    end
    set $n = n + 1$
until $n \geqslant N$;
```
/*at this stage, none of the elements y has been
  returned during the for-loop */
```
return FAIL

Algorithm 2: A BASICSIFT algorithm using random search.

where $p = p(H, K, L)$ and ϱ, ξ and $\nu(e)$ are upper bounds for the costs of a group operation in G, a random selection from L, and one run of the procedure ISMEMBER$(\cdot, e)$, respectively, where $e = 0$ if ISMEMBER is deterministic, and $e = \varepsilon p(H, K, L)/(2 - 2p(H, K, L))$ otherwise.

Proof. If a group element y is returned, then, by (3), $Hy \subseteq HL \subseteq H$.

Let E denote the event that 'the output of the procedure is either FAIL or an element y with $xy \notin K$'. We are required to show that $\text{Prob}(E|x \in H) \leqslant \varepsilon$. Suppose that $x \in H$. For $i = 0, \ldots, N - 1$, let E_i denote the event 'the $(i + 1)$th execution of the procedure RANDOMELEMENT occurs'; let y_i denote the element y returned by the $(i + 1)$th execution of RANDOMELEMENT, and let z_i denote the result returned by the call to ISMEMBER(xy_i, e). If E_i does not occur for some i, then the values of y_i and z_i are undefined. The event E_i is the disjoint union of the following three events:

$$K_i : E_i \text{ and } xy_i \in K;$$
$$F_i : E_i \text{ and } xy_i \notin K \text{ and } z_i = \text{FALSE};$$
$$T_i : E_i \text{ and } xy_i \notin K \text{ and } z_i = \text{TRUE}.$$

Note that E_i occurs if and only if, for each $j < i$, the event E_j has occurred and $z_j = $ FALSE; that is to say, $E_i = F_0 \cap \cdots \cap F_{i-1}$. Similarly, given $x \in H$, the event E occurs if and only if either $F_0 \cap F_1 \cap \cdots \cap F_{N-1}$ occurs, or if, for some i, each of $E_1, \ldots, E_i$ occurs, $xy_i \notin K$ and $z_i = $ TRUE.

Suppose now that $x \in H$, and let $y \in L$ be such that $xy \notin K$. Then by (3), $xy \in HL \subseteq H$, and hence $xy \in H \setminus K$. By the definition of the membership test condition, the conditional

probability e_0 that the returned value of IsMember(xy, e) is True, given that $xy \in H \setminus K$, satisfies $0 \leqslant e_0 \leqslant e$.

Let p denote the sifting parameter $p(H, K, L)$. Since we are making independent uniform random selections, we see, for each $i \leqslant N - 1$, that the probability Prob$(K_i|E_i)$ is independent of i, and also that

$$\text{Prob}(K_i|E_i) \geqslant \frac{|xL \cap K|}{|L|} \geqslant p.$$

Set $p_0 = \text{Prob}(K_i|E_i)$. Then, using the rule

$$\text{Prob}(A \cap B|C) = \text{Prob}(A|B \cap C)\,\text{Prob}(B|C),$$

we have

$$\text{Prob}(F_i|E_i) = \text{Prob}(xy_i \notin K|E_i) \cdot \text{Prob}(z_i = \text{False}|E_i \text{ and } xy_i \notin K)$$
$$= (1 - p_0)(1 - e_0)$$

with e_0 as defined above; similarly, $\text{Prob}(T_i|E_i) = (1 - p_0)e_0$.

The procedure finishes when processing the ith random element y_i if it has not finished while processing y_j for any $j < i$, and either K_i or T_i occurs. In this situation, if K_i occurs, then by the requirements of the membership test condition, the procedure will return y_i with $xy_i \in K$; similarly, if T_i occurs, then again the procedure will return y_i, but this time with $xy_i \notin K$. Thus the procedure returns the element y_i with $xy_i \notin K$ (for a particular value of i) if and only if $F_0 \cap \cdots \cap F_{i-1} \cap T_i = T_i$ occurs, and

$$\text{Prob}(T_i) = e_0(1 - p_0)((1 - p_0)(1 - e_0))^i.$$

It follows that the procedure returns an element $y \in L$ with $xy \notin K$ if and only if T_i occurs for some $i = 0, \ldots, N - 1$, and the probability of this is

$$\sum_{i=0}^{N-1} e_0(1 - p_0)^{i+1}(1 - e_0)^i = e_0(1 - p_0)\frac{1 - (1 - p_0)^N(1 - e_0)^N}{1 - (1 - p_0)(1 - e_0)}$$
$$\leqslant \frac{(1 - p_0)e_0}{p_0},$$

since $1 - (1 - p_0)(1 - e_0) = p_0 + (1 - p_0)e_0 \geqslant p_0$. Finally, the procedure returns Fail if and only if the event $F_0 \cap F_1 \cap \cdots \cap F_{N-1}$ occurs; the probability of this is $(1 - p_0)^N(1 - e_0)^N$.

We derive the required estimates of these probabilities as follows. Note that, since $p \leqslant p_0$ and $0 \leqslant e_0 \leqslant e$, we have

$$\frac{(1 - p_0)e_0}{p_0} = (p_0^{-1} - 1)e_0 \leqslant (p^{-1} - 1)e,$$

and this is 0 if $e = 0$, and is $\varepsilon/2$ otherwise. Hence, the probability that the procedure returns an element $y \in L$, with $Hy \subseteq H$ and $xy \notin K$, is 0 if IsMember is deterministic, and is at most $\varepsilon/2$ otherwise. Similarly, the probability that the procedure returns Fail is

$$(1 - p_0)^N(1 - e_0)^N \leqslant (1 - p)^N \leqslant \frac{\varepsilon}{\delta},$$

by the definition of N, where $\delta = 1$ if IsMember is deterministic, and $\delta = 2$ otherwise. Thus $(G_0, H, K, \text{BasicSift})$ satisfies the basic sift condition in G.

Finally we estimate the cost. For each run of the **repeat** loop, first we select a random element of L at a cost of at most ξ. Then we perform a group operation to compute xy,

and we run IsMember(xy, e) at a cost of at most $\varrho + \nu(e)$, where $e = 0$ if IsMember is deterministic, and $e = \varepsilon p/(2(1 - p))$ otherwise. The number of runs of the loop is at most N and, by Lemma 3.2, N is $O(\log(\varepsilon^{-1})p^{-1})$. Thus the upper bound for the cost is proved. (Note that, for $\varepsilon < 1/2$ we have $\varepsilon p/(2(1 - p)) < 1/2$ also.) □

As already explained before Theorem 5.3, we often work with sifting triples (H, K, L) in which L is a subgroup of G_0. Usually, there will be another subgroup $L' < L$, which is used to define K, and we have $KL' \subseteq K$. In this situation the following concept applies.

DEFINITION 5.4. Suppose that G is a finite group, and that L and L' are subgroups of G. A non-empty subset S of L is said to be *left L'-uniform* if S has the same number of elements in each of the left L'-cosets in L. In other words, $|S \cap \ell L'|$ is constant for all $\ell \in L$.

A left L'-uniform subset in L must contain a left transversal for L' in L. Notice that L is left $\{1_G\}$-uniform, and more generally, if L' is a subgroup, then any left transversal for L' in L is left L'-uniform. As will become clear in the next lemma, L'-uniform sets S have 'nice' properties with respect to the calculation of probabilities. In certain cases, we need to consider sifting triples (H, K, S) in which S is a left L'-uniform subset in some subgroup L for which (H, K, L) is also a sifting triple. We show that in such cases the sifting parameter $p(H, K, S)$ is independent of the subgroup L' and the left L'-uniform subset S, and depends only on the subgroup L.

LEMMA 5.5. *Let (H, K, L) be a sifting triple in which L is a subgroup, let L' be a subgroup of L with $KL' \subseteq K$, and let S be a left L'-uniform subset of L. Then (H, K, S) is also a sifting triple and $p(H, K, S) = p(H, K, L)$.*

Proof. Since $HL \subseteq H$ and $S \subseteq L$, it follows that $HS \subseteq H$. Let $h \in H$. We shall show that $|hS \cap K|/|S| = |hL \cap K|/|L|$. The result will then follow. By (3), $hL \cap K \neq \varnothing$. Note that, since L' is a subgroup of L, and since S is left L'-uniform, it follows that $L = SL'$, and $LL' = L$. In addition, we have $KL' = K$. Thus Lemma 3.3 implies that $(hL \cap K)L' = hL \cap K$, and in particular, $hL \cap K$ is a union of r left L'-cosets, for some $r > 0$. Each of these cosets is contained in $hL = hSL'$ and hence is of the form hsL' for some $s \in S$. Thus $hL \cap K = \bigcup_{i=1}^{r} hs_i L'$ for some $s_1, \ldots, s_r \in S$.

Further, since S is left L'-uniform, the size $q = |s_i L' \cap S|$ is independent of i. Moreover, for each $i \leqslant r$, $hs_i L' \cap hS = h(s_i L' \cap S)$, and since $hS \subseteq hL$ it follows that

$$hS \cap K = (hL \cap K) \cap hS = \bigcup_{i=1}^{r} (hs_i L' \cap hS) = \bigcup_{i=1}^{r} h(s_i L' \cap S),$$

and therefore $|hS \cap K| = rq$. On the other hand, $hL \cap K = \bigcup_{i=1}^{r} hs_i L'$ has size $r|L'|$. Since S has exactly q elements in each of the left L'-cosets in L, we have $|S| = q|L : L'|$, and hence

$$\frac{|hL \cap K|}{|L|} = \frac{r|L'|}{|L|} = \frac{rq}{|S|} = \frac{|hS \cap K|}{|S|},$$

proving the claim. □

5.2. *A* BASICSIFT *algorithm using a stored transversal*

We turn now to a second general approach to designing a 4-tuple that satisfies the basic sift condition. This algorithm is defined for the case when we have a sifting triple (H, K, L) and a subgroup $L' \leqslant L$ as in Lemma 5.5. Unlike Algorithm 2, where we choose elements

---

Algorithm 3: BASICSIFTCOSETREPS
```
/*See Theorem 5.6 for notation */
```

Input: (g, ε) where $g \in G$, and $0 \leqslant \varepsilon < 1/2$;
```
/* ε = 0 if and only if ISMEMBER is deterministic */
```
Output: y, where either $y = $ FAIL, or $y \in S$;
set
$$e = \begin{cases} 0 & \text{if ISMEMBER is deterministic,} \\ \min\left\{\varepsilon \cdot \dfrac{n+1}{k-n}, \dfrac{1}{3}\right\} & \text{otherwise, where } k = |S|, \ n = \min_{h \in H} |hS \cap K|; \end{cases}$$
set $T = S$;
for $i = 1$ **to** k **do**
    set $y = $ RANDOMELEMENT(T);
    if ISMEMBER $(gy, e) = $ TRUE **then**
        return y
    end
    set $T = T \setminus \{y\}$;
end
```
/* we reach this stage only if g ∉ H, because otherwise
   one of the ISMEMBER tests must have returned TRUE */
```
return FAIL

Algorithm 3: A BASICSIFT algorithm using a left transversal of L' in L.

of L at random, Algorithm 3 deterministically tests every element of a complete set S of left coset representatives calculated beforehand. Thereby we turn the random search above into a deterministic, exhaustive search. As will be explained below, this can significantly reduce the expected value of the runtime.

We use Algorithm 3 when the index of L' in L, and thus the size of S, is small enough to allow S to be stored completely. We still allow the use of randomised or deterministic ISMEMBER methods. In the latter case, the whole basic sift procedure is deterministic.

We would like to draw attention to a little trick that we use to simplify the analysis of the error probability of Algorithm 3. We artificially introduce a randomly chosen order in which the coset representatives are tried. This makes the analysis less dependent on the input group element.

THEOREM 5.6. *Suppose that $(G_0, H, K, $ ISMEMBER$)$ satisfies the membership test condition in a group G. Assume further that L is a subgroup of G_0, such that (H, K, L) is a sifting triple, that $L' < L$ with $KL' = K$, and that $S = \{s_1, \ldots, s_k\}$ is a left transversal of L' in L. If, for any $T \subseteq S$, RANDOMELEMENT(T) returns uniformly distributed, independent random elements of T, and BASICSIFT is Algorithm 3, then the 4-tuple $(G_0, H, K, $ BASICSIFT$)$ satisfies the basic sift condition in G.*

The cost of executing BASICSIFTCOSETREPS$(\cdot, \varepsilon)$ is less than $k \cdot (\xi_S + \varrho + \nu(e))$, where ξ_S is an upper bound for the cost of selecting a random element from a subset of S, and ϱ and $\nu(e)$ are upper bounds for the costs of, respectively, a group operation in G, and one run of the procedure ISMEMBER$(\cdot, e)$. Here, $e = 0$ if ISMEMBER is deterministic, and $e = \min\{\varepsilon(n+1)/(k-n), 1/3\}$ otherwise, where $n = \min_{h \in H} |hS \cap K|$.

Proof. We remark, first, that for every $g \in H$ there is an element $l \in L$ such that $gl \in K$ by hypothesis (3). As S is a left transversal for L' in L, there are $s \in S$ and $l' \in L'$ such that $l = sl'$. Now $gsl' \in K$, and so $gs \in KL' = K$. Therefore, if $g \in H$, then Algorithm 3 cannot return FAIL, as the IsMember test is one-sided Monte Carlo. Also, this argument proves all the statements in the theorem in the case where IsMember is deterministic.

Thus from now on we will assume that IsMember is not deterministic, and therefore that $0 < \varepsilon < 1/2$, and hence e is non-zero.

As $HL = H$, the set H is a union of left L-cosets, and, a fortiori, also a union of left L'-cosets. Analogously, $KL' = K$ means that K is a union of left L'-cosets, and, of course, so is $gL \cap K$. For any given g, the algorithm looks for a random element y in $S \subset L$ such that $gy \in K$; in other words, it searches the coset gL for elements of K. Thus, the number of elements $s \in S$ with $gs \in K$ is equal to the number of left L'-cosets contained in $gL \cap K$. Let $g \in H$. As, by Lemma 5.5, $p(H, K, S) = p(H, K, L)$ and $|gL \cap K|/|L| = |gL \cap K|/(k|L'|)$, we obtain

$$|gS \cap K| = |S| \frac{|gS \cap K|}{|S|} \geqslant |S| \min_{h \in H} \frac{|hS \cap K|}{|S|} = kp(H, K, S) = kp(H, K, L).$$

Let E denote the event 'the procedure returns $y \in S$ with $gy \notin K$'. To check the basic sift condition for Algorithm 3 in the case of a randomised IsMember test, we have to show that $\text{Prob}(E | g \in H) \leqslant \varepsilon$.

Suppose now that $g \in H$. For $i = 1, \ldots, k$, let E_i denote the event: 'the ith execution of the procedure RandomElement occurs'; let y_i denote the element y returned by the ith execution of the procedure RandomElement, and let z_i denote the result returned by the call to IsMember(gy_i, e) (for steps i that do not happen, y_i and z_i are undefined).

Then E_i is the disjoint union of the following three events:

$$K_i : E_i \text{ and } gy_i \in K;$$
$$F_i : E_i \text{ and } gy_i \notin K \text{ and } z_i = \text{FALSE};$$
$$T_i : E_i \text{ and } gy_i \notin K \text{ and } z_i = \text{TRUE}.$$

Note that E_i occurs if and only if, for each $j < i$, the event E_j has occurred and $z_j = \text{FALSE}$; that is to say, $E_i = F_1 \cap \cdots \cap F_{i-1}$. Similarly, given $g \in H$, the event E occurs if and only if, for some i, each of $E_1, \ldots, E_i$ occurs, $gy_i \notin K$, and $z_i = \text{TRUE}$. Thus, given $g \in H$, the event E occurs if, and only if, $F_1 \cap \cdots \cap F_{i-1} \cap T_i = T_i$ occurs for some i with $1 \leqslant i \leqslant k$.

Since in step i we choose y_i only among those coset representatives that have not been tried before, and we reach step i only if $gy_j \notin K$ for $1 \leqslant j < i$, the probability $\text{Prob}(gy_i \notin K | E_i)$ is not independent of i. In fact,

$$\text{Prob}(gy_i \notin K | E_i) = \frac{k + 1 - i - n_g}{k + 1 - i}$$

where $n_g = |gS \cap K|$, as in step i there are $k + 1 - i$ coset representatives in the set T, of which $k + 1 - i - n_g$ do not multiply g into K. For $i > k - n_g$, the probability $\text{Prob}(gy_i \notin K \mid E_i) = 0$. Now

$$\text{Prob}(F_i | E_i) = \text{Prob}(gy_i \notin K | E_i) \cdot \text{Prob}(z_i = \text{FALSE} | gy_i \notin K \text{ and } E_i)$$

and so

$$\text{Prob}(F_i | E_i) \leqslant \frac{k + 1 - i - n_g}{k + 1 - i}.$$

Similarly, we have

$$\text{Prob}(T_i|E_i) \leqslant \frac{k+1-i-n_g}{k+1-i} \cdot e.$$

As in the proof of Theorem 5.3, Algorithm 3 finishes in step i, if it has not finished in an earlier step and K_i or T_i occurs. In this situation, if K_i occurs, then the procedure will return y_i with $gy_i \in K$, which is a correct result. Therefore, an error produced by step i occurs exactly in the event T_i, and

$$\text{Prob}(T_i) \leqslant \left(\prod_{j=1}^{i} \frac{k+1-j-n_g}{k+1-j} \right) \cdot e.$$

Moreover, no error can possibly occur in step i for $i > k - n_g$.

Therefore, for an input (g, ε) with $g \in H$, the total probability that Algorithm 3 returns an element $y \in S$ with $gy \notin K$ is at most

$$\sum_{i=1}^{k-n_g} \left(\prod_{j=1}^{i} \frac{k+1-j-n_g}{k+1-j} \right) \cdot e.$$

Note that, for $i = 1, \ldots, k - n_g$,

$$\prod_{j=1}^{i} \frac{k+1-j-n_g}{k+1-j} = \frac{(k-n_g)(k-n_g-1)\cdots(k-n_g-i+1)}{k(k-1)\cdots(k-i+1)}$$

$$= \frac{(k-i)!}{k!} \cdot \frac{(k-n_g)!}{(k-i-n_g)!}.$$

Hence

$$\sum_{i=1}^{k-n_g} \left(\prod_{j=1}^{i} \frac{k+1-j-n_g}{k+1-j} \right) = \sum_{i=1}^{k-n_g} \frac{(k-i)!}{k!} \cdot \frac{(k-n_g)!}{(k-i-n_g)!} \cdot \frac{n_g!}{n_g!}$$

$$= \binom{k}{n_g}^{-1} \cdot \sum_{i=1}^{k-n_g} \binom{k-i}{n_g}.$$

We can simplify the sum further by repeated use of the well-known summation formula for binomial coefficients:

$$\binom{a}{b} + \binom{a}{b-1} = \binom{a+1}{b}.$$

The last summand (with $i = k - n_g$) is equal to

$$\binom{n_g}{n_g} = 1 = \binom{n_g+1}{n_g+1}.$$

In the latter form, it can be added to the second-last summand, resulting in $\binom{n_g+2}{n_g+1}$. This can be repeated until the first summand, thereby proving that

$$\sum_{i=1}^{k-n_g} \binom{k-i}{n_g} = \binom{k}{n_g+1}.$$

This, however, implies that the total probability of an error is at most

$$\binom{k}{n_g}^{-1} \cdot \binom{k}{n_g + 1} \cdot e = \frac{n_g! \cdot (k - n_g)!}{k!} \cdot \frac{k!}{(n_g + 1)! \cdot (k - n_g - 1)!} \cdot e = \frac{k - n_g}{n_g + 1} \cdot e.$$

Thus, as $n \leqslant n_g$, for an arbitrary element $g \in H$, the error probability is bounded by

$$\frac{k - n}{n + 1} \cdot e \leqslant \varepsilon.$$

As for the cost, the loop terminates at the latest after k steps, each of which has a random element selection from T, one group multiplication for computing gy_i, and one call to IsMember. □

Our hypotheses in Theorem 5.6 imply that S is L'-uniform. However, since we want to store S completely, there is no point in choosing left L'-uniform sets with more than one element in each left L'-coset of L.

5.3. *Comments on and comparison of Algorithms 2 and 3*

To compare Algorithms 2 and 3, assume that L is a subgroup, and that we want to sift from a set H with $HL = H$ down to a set K with $KL' = K$, and that $L' < L$ with $[L : L'] = k$. Then we can either use Algorithm 2, or use Algorithm 3 with S being a left transversal of L' in L. Recall that $p(H, K, L) = p(H, K, S) = p$, say (see Lemma 5.5). Let k denote the index $|L : L'|$, and let n denote $\min_{h \in H} |hS \cap K| = pk$. In the second case, we have to calculate and store S beforehand. In Algorithm 3, once we compute that a random element y does not multiply g into K, y cannot be selected again by a subsequent call of RandomElement. Therefore we expect that Algorithm 3 will perform better than Algorithm 2 in this situation.

In Algorithm 2 the bound for the error probability in all calls of the IsMember test is $e_1 = \varepsilon p/(2 - 2p) = \varepsilon n/(2(k - n))$ (recall that $p = n/k$), whereas in Algorithm 3 the bound for the error probability for the IsMember calls is $e_2 = \varepsilon(n + 1)/(k - n)$ (at least when ε is not too big, so that e_2 is not defined to be $1/3$), which is a little bit more than $2e_1$. Thus, due to the deterministic nature of the choice of y in Algorithm 3, we can afford bigger error bounds for the IsMember tests. Further, the expected number of steps in Algorithm 2 is $1/p$ (geometric distribution), which is k/n as $p = n/k$. The expected number of steps in Algorithm 3 is $(k + 1)/(n + 1)$.

These calculations suggest that, whenever it is possible to store all elements of S, Algorithm 3 should be preferred over Algorithm 2.

If the IsMember test is deterministic and happens to work not only for elements of H but also for arbitrary elements of HS, then one can dispense altogether with the hypothesis that $HS \subseteq H$, and apply Algorithm 3 verbatim for any set $S \subseteq G$ satisfying $hS \cap K \neq \varnothing$ for all $h \in H$. In this case, Algorithm 3 will be a fully deterministic algorithm with a guaranteed, finite runtime of at most $|S|$ steps.

6. BasicSift: *with special subsets H and K*

In this section we describe a rather general situation where the conditions in (3) are guaranteed to hold. The conditions on the subsets H and K of the finite group G are as follows:

$$H = CL = C'L, \ K = C'L', \ \text{where } L' < L \leqslant G,$$
$$C, C' \subseteq G, \ \text{with } C, C' \neq \varnothing. \tag{4}$$

Under these conditions we also derive a new expression for the sifting parameter $p(H, K, L)$ required for Algorithm 2 and Theorem 5.3.

PROPOSITION 6.1. *Let G, L, L', C, C', H and K be as above, so that* (4) *holds. Then $K \subseteq H$, and if $H \neq K$ then (H, K, L) is a sifting triple. Further,*

$$p(H, K, L) = \min_{y \in \mathcal{Y}} \frac{|yL \cap K|}{|L|} = \min_{y \in \mathcal{Y}} \frac{|(yL \cap C')L'|}{|L|},$$

where $\mathcal{Y}$ is a set of representatives in C' for the left L-cosets contained in H.

Proof. Since $L' \subseteq L$, we have $C'L' \subseteq C'L = CL$; that is, $K \subseteq H$. Note that, since $1 \in L'$, we have $C' \subseteq K$ and $C \subseteq H$.

Suppose now that $K \neq H$. Since $H = CL$ and L is a subgroup, it follows that $HL \subseteq H$. Let $y \in H$. To complete the proof of (3), we need to show that $yL \cap K$ is non-empty. Since $y \in H$ and $H = CL = C'L$, we have $y = ck$ where $c \in C'$ and $k \in L$, and hence $c = yk^{-1}$ and $c \in yL \cap C'$. As $C' \subseteq K$, we obtain $c \in yL \cap K$. Thus $yL \cap K \neq \varnothing$.

Now it remains only to show that the assertion in the displayed line of the proposition is valid. It follows from (4) that, for $y \in H$, $yL \cap K = yL \cap (C'L') = (yL \cap C')L'$, by Dedekind's modular law (Lemma 3.3). Hence, for all $y \in H$, we have

$$\frac{|yL \cap K|}{|L|} = \frac{|(yL \cap C')L'|}{|L|}.$$

Suppose that $y \in H$ and $y = ck$ where $c \in C'$ and $k \in L$. Then $yL \cap K = cL \cap K$, and so the minimum value of $|yL \cap K|/|L|$ over all $y \in H$ is equal to the minimum value of $|cL \cap K|/|L|$ over all $c \in \mathcal{Y}$. The displayed assertion follows. □

We will apply Algorithm 2 with H and K as in (4) in the following context: G_0 is a subgroup of a finite group G, the group G_0 has a descending subgroup chain

$$G_0 = L_0 > L_1 > \cdots > L_k = \{1\}, \tag{5}$$

and also has a sequence of non-empty subsets

$$C_0, C_1, \ldots, C_k \quad \text{such that} \quad C_{i+1}L_i = C_iL_i \text{ for all } i < k. \tag{6}$$

Thus (4) holds for $(H, K) = (C_iL_i, C_{i+1}L_{i+1})$ for each $i < k$. By Proposition 6.1, we have a descending chain

$$G_0 = C_0L_0 \supseteq C_1L_1 \supseteq \cdots \supseteq C_kL_k = C_k, \tag{7}$$

and by Proposition 6.1, Algorithm 2 applies to each of the pairs $(C_iL_i, C_{i+1}L_{i+1})$ such that $C_iL_i \neq C_{i+1}L_{i+1}$ ($0 \leqslant i < k$). Thus if, for $i = 0, \ldots, k-1$, the 4-tuple $(G_0, C_iL_i, C_{i+1}L_{i+1}, \text{ISMEMBER}_i)$ satisfies the membership test condition in G for some algorithm ISMEMBER$_i$, and if we have an algorithm BASICSIFT$_k$ such that the 4-tuple $(G_0, C_k, \{1\}, \text{BASICSIFT}_k)$ satisfies the basic sift condition in G, then we may use the procedures BASICSIFT$_i$ in Algorithm 1. If $|C_k|$ is small, BASICSIFT$_k$ may simply test each member of C_k for equality with the input element (if $1 \notin C_k$), or its inverse (if $1 \in C_k$). The next two sections offer some possibilities for these procedures that have been found to be effective for computing with some of the sporadic simple groups.

7. IsMember *using conjugates*

In this section we apply the theory developed in Sections 5 and 6, especially in Section 6, to sift an element down a subgroup chain such as (5) by making use of an auxiliary subset sequence. This application uses conjugates of an element a with the following property:

$$a \in L_{k-1} \setminus \{1\} \text{ such that, for each } i = 0, \dots, k-2, \text{ each} \\ L_i\text{-conjugacy class in } a^{G_0} \cap L_i \text{ intersects } L_{i+1} \text{ non-trivially.} \tag{8}$$

We construct an associated subset sequence (6) recursively as follows. The first subset is $C_0 = C_{G_0}(a)\mathcal{T}_0$, where $\mathcal{T}_0 = \{1\}$. Consider a typical link in the chain (5), say $L_i > L_{i+1}$ for $i \leqslant k-2$, and suppose that we have already constructed the subset C_i corresponding to L_i, and C_i is of the form $C_i = C_{G_0}(a)\mathcal{T}_i$, where $\{a^y \mid y \in \mathcal{T}_i\}$ is a set of L_i-conjugacy class representatives in $a^{G_0} \cap L_i$. Then

$$a^{G_0} \cap L_{i+1} = \bigcup_{y \in \mathcal{T}_i} (a^{yL_i} \cap L_{i+1}),$$

and by condition (8), each $a^{yL_i} \cap L_{i+1}$ is non-empty. For each $y \in \mathcal{T}_i$, choose $\mathcal{U}(y) \subset L_i$ such that $\{a^{yu} \mid u \in \mathcal{U}(y)\}$ is a set of representatives for the L_{i+1}-conjugacy classes in $a^{yL_i} \cap L_{i+1}$. Define $\mathcal{T}_{i+1} = \bigcup_{y \in \mathcal{T}_i} y\mathcal{U}(y)$, and define the subset C_{i+1} corresponding to L_{i+1} by $C_{i+1} = C_{G_0}(a)\mathcal{T}_{i+1}$. In addition, set $C_k = \{1\}$.

We prove that (4) holds, and we also derive two expressions for the sifting parameter $p(H, K, L)$ required for Algorithm 2 and Theorem 5.3. The first expression shows that $p(H, K, L)$ is a ratio of the sizes of two special subsets of conjugates of the element a, while the second expression provides a means of computing $p(H, K, L)$ from the orders of various centraliser subgroups.

PROPOSITION 7.1. *Suppose that G, G_0, a, L_i, L_{i+1}, C_i, C_{i+1}, $\mathcal{T}_i$, $\mathcal{T}_{i+1}$, and the $\mathcal{U}(x)$, for $x \in \mathcal{T}_i$, are as at the beginning of this section, and set $H = C_i L_i$, $K = C_{i+1}L_{i+1}$, $L = L_i$, $L' = L_{i+1}$, $C = C_i$ and $C' = C_{i+1}$. Then $\mathcal{T}_{i+1}L_i = \mathcal{T}_i L_i$ and (4) holds, and if also $H \neq K$, then (H, K, L) is a sifting triple. Further,*

$$p(H, K, L) = \min_{x \in \mathcal{T}_i} \frac{|a^{xL_i} \cap L_{i+1}|}{|a^{xL_i}|} = \frac{1}{|L : L'|} \min_{x \in \mathcal{T}_i} \left\{ |C_L(a^x)| \sum_{u \in \mathcal{U}(x)} \frac{1}{|C_{L'}(a^{xu})|} \right\}.$$

Proof. By the definition of $\mathcal{T}_{i+1}$, we know that $\mathcal{T}_{i+1} \subseteq \mathcal{T}_i L_i$. Also, since (8) holds, for each $x \in \mathcal{T}_i$ there exists $k \in L_i$ such that $xk \in \mathcal{T}_{i+1}$. Thus $\mathcal{T}_i \subseteq \mathcal{T}_{i+1}L_i$, and so, since L_i is a subgroup, we have

$$\mathcal{T}_i L_i \subseteq (\mathcal{T}_{i+1}L_i)L_i = \mathcal{T}_{i+1}L_i \subseteq (\mathcal{T}_i L_i)L_i = \mathcal{T}_i L_i.$$

Hence $\mathcal{T}_{i+1}L_i = \mathcal{T}_i L_i$. To prove (4), it is sufficient to prove that $H = C'L = C_{i+1}L_i$. From the definition of H we have

$$H = C_i L_i = C_{G_0}(a)\mathcal{T}_i L_i = C_{G_0}(a)\mathcal{T}_{i+1}L_i = C_{i+1}L_i = C'L.$$

Thus (4) holds. Moreover, if $H \neq K$, then, by Proposition 6.1, (H, K, L) is a sifting triple.

It remains to show that the value of the sifting parameter $p(H, K, L)$ is as claimed. Suppose that $h \in H$, and that $h = cxk$ with $c \in C_{G_0}(a)$, $x \in \mathcal{T}_i$, and $k \in L_i$. We claim that $|hL_i \cap K| = |(xL_i \cap C_{i+1})L_{i+1}|$. As $k \in L_i$, we certainly have $hL_i \cap K = cxL_i \cap K$. An easy calculation shows that $cxL_i \cap C_{G_0}(a)\mathcal{T}_{i+1}L_{i+1} = c(xL_i \cap C_{G_0}(a)\mathcal{T}_{i+1}L_{i+1})$, and so

$|cxL_i \cap C_{i+1}L_{i+1}| = |xL_i \cap C_{i+1}L_{i+1}|$. Therefore $|hL_i \cap K| = |xL_i \cap K|$. Finally, by Dedekind's modular law (Lemma 3.3, which applies since $(xL_i)L_{i+1} \subseteq xL_i$), we obtain

$$xL_i \cap K = xL_i \cap C_{i+1}L_{i+1} = (xL_i \cap C_{i+1})L_{i+1},$$

proving our claim.

Next we show that $xL_i \cap C_{i+1} = xC_{L_i}(a^x)\mathcal{U}(x)$, with $\mathcal{U}(x)$ as defined before Proposition 7.1 (recall that $x \in \mathcal{T}_i$). Let $y \in xL_i \cap C_{i+1}$, so that $y = xk$ for some $k \in L_i$ and $xk \in C_{i+1}$. Since $C_{i+1} = C_{G_0}(a)\mathcal{T}_{i+1}$, it follows that $a^{xk} \in a^{\mathcal{T}_{i+1}} \cap a^{xL_i}$. By the definition of $\mathcal{T}_{i+1}$, there is some $u \in \mathcal{U}(x)$ such that $a^{xk} = a^{xu}$, and so $k \in C_{L_i}(a^x)u$. Therefore $xk \in xC_{L_i}(a^x)u$, and we find that $y = xk \in xC_{L_i}(a^x)\mathcal{U}(x)$. Conversely, consider $y = xcu$, where $c \in C_{L_i}(a^x)$ and $u \in \mathcal{U}(x)$. As $\mathcal{U}(x) \subseteq L_i$, we have $y = xcu \in xL_i$. Further, $a^{xcu} = a^{xu} \in a^{x\mathcal{U}(x)} \subseteq a^{\mathcal{T}_{i+1}}$. Thus $y = xcu \in C_{G_0}(a)\mathcal{T}_{i+1} = C_{i+1}$. Therefore our claim is proved.

Putting the calculations in the last two paragraphs together, we have shown, for $h = cxk$ with $c \in C_{G_0}(a)$, $x \in \mathcal{T}_i$ and $k \in L_i$, that $|hL_i \cap K| = |xC_{L_i}(a^x)\mathcal{U}(x)L_{i+1}|$. Now we calculate the size of $xC_{L_i}(a^x)\mathcal{U}(x)L_{i+1}$. We first observe that $xC_{L_i}(a^x)\mathcal{U}(x)L_{i+1}$ is a union of left L_{i+1}-cosets, and hence it suffices to compute the number of such cosets contained in $xC_{L_i}(a^x)\mathcal{U}(x)L_{i+1}$. If u_1 and u_2 are distinct elements of $\mathcal{U}(x)$, then $a^{xC_{L_i}(a^x)u_1L_{i+1}} = a^{xu_1L_{i+1}}$ and $a^{xC_{L_i}(a^x)u_2L_{i+1}} = a^{xu_2L_{i+1}}$, and so it follows from the definition of $\mathcal{U}(x)$ that $a^{xC_{L_i}(a^x)u_1L_{i+1}}$ and $a^{xC_{L_i}(a^x)u_2L_{i+1}}$ are distinct conjugacy classes in L_{i+1}. Thus $xC_{L_i}(a^x)u_1L_{i+1}$ and $xC_{L_i}(a^x)u_2L_{i+1}$ are disjoint. Therefore $xC_{L_i}(a^x)\mathcal{U}(x)L_{i+1}$ is the disjoint union, over all $u \in \mathcal{U}(x)$, of $xC_{L_i}(a^x)uL_{i+1}$. Let $c_1, c_2 \in C_{L_i}(a^x)$. Then $xc_1uL_{i+1} = xc_2uL_{i+1}$ if and only if $c_2^{-1}c_1 \in uL_{i+1}u^{-1}$. Thus the number of left L_{i+1}-cosets in $xC_{L_i}(a^x)uL_{i+1}$ is $|C_{L_i}(a^x)|/|C_{uL_{i+1}u^{-1}}(a^x)| = |C_{L_i}(a^x)|/|C_{L_{i+1}}(a^{xu})|$. Hence, the definition of $\mathcal{U}(x)$ implies that

$$
\begin{aligned}
|hL_i \cap K| &= |xC_{L_i}(a^x)\mathcal{U}(x)L_{i+1}| \\
&= \sum_{u \in \mathcal{U}(x)} |xC_{L_i}(a^x)uL_{i+1}| \\
&= \sum_{u \in \mathcal{U}(x)} \frac{|C_{L_i}(a^x)| \cdot |L_{i+1}|}{|C_{L_{i+1}}(a^{xu})|} \\
&= |C_{L_i}(a^x)| \sum_{u \in \mathcal{U}(x)} \frac{|L_{i+1}|}{|C_{L_{i+1}}(a^{xu})|} \\
&= |C_{L_i}(a^x)| \sum_{u \in \mathcal{U}(x)} |(a^{xu})^{L_{i+1}}| \\
&= |C_{L_i}(a^x)| \cdot |a^{xL_i} \cap L_{i+1}|.
\end{aligned}
$$

Thus

$$\frac{|hL_i \cap K|}{|L_i|} = \frac{|C_{L_i}(a^x)| \cdot |a^{xL_i} \cap L_{i+1}|}{|L_i|} = \frac{|a^{xL_i} \cap L_{i+1}|}{|a^{xL_i}|}$$

and also

$$\frac{|hL_i \cap K|}{|L_i|} = \frac{|C_{L_i}(a^x)|}{|L_i : L_{i+1}|} \sum_{u \in \mathcal{U}(x)} \frac{1}{|C_{L_{i+1}}(a^{xu})|}.$$

Therefore we see that the displayed assertions for the sifting parameter also hold. $\square$

Algorithm 4: IsMEMBERCONJUGATES
/*see Lemmas 7.2 and 7.3 for notation */

Input: (x, e) where $x \in G$, and $e = 0$ if IsMEMBER is deterministic, and $0 < e < 1/2$ otherwise;
Output: TRUE or FALSE;
return IsMEMBER(a^x, e)

Algorithm 4: An IsMEMBER algorithm for subsets.

The main benefit of working with conjugates is that, using the notation of Proposition 7.1, membership of x in H or K is equivalent to membership of a^x in L_i or L_{i+1}, respectively; see Lemma 7.2. It is often easier to test whether a random conjugate of a known element lies in a subgroup, than to test membership of a random element in a subgroup or subset. This is true in particular if we have detailed information about subgroups of L_i or L_{i+1} generated by two a-conjugates.

LEMMA 7.2. *Let G, G_0, a, L_i, L_{i+1}, C_i, C_{i+1}, $\mathcal{T}_i$ and $\mathcal{T}_{i+1}$, be as in Proposition 7.1, set $H = C_i L_i$ and $K = C_{i+1} L_{i+1}$, and let $x \in G$.*

(a) *The element $x \in H$ if and only if $a^x \in L_i$; similarly, $x \in K$ if and only if $a^x \in L_{i+1}$.*

(b) *If $(G_0, L_i, L_{i+1}, \text{IsMEMBER})$ satisfies the membership test condition in G, for some algorithm IsMEMBER, then so does $(G_0, H, K, \text{IsMEMBERCONJUGATES})$ where the algorithm IsMEMBERCONJUGATES is given by Algorithm 4.*

Proof. It follows from the definition of $\mathcal{T}_i$ that $a^{\mathcal{T}_i L_i} = a^{G_0} \cap L_i$. The first assertion in part (a) is then obvious, and the second follows similarly.

To prove part (b), recall the second assertion of part (a), namely that $x \in K$ if and only if $a^x \in L_{i+1}$. If this condition holds, then the membership test condition (see Definition 5.1) on IsMEMBER implies that IsMEMBER(a^x, e) = TRUE, and hence we obtain IsMEMBERCONJUGATES(x, e) = TRUE. Also, by part (a), $x \in H \setminus K$ if and only if $a^x \in L_i \setminus L_{i+1}$. By the membership test condition on IsMEMBER, we have

$$\text{Prob(output of IsMEMBER is TRUE}|x \in H \setminus K) \leqslant e$$

and hence by the 'definition' of IsMEMBERCONJUGATES in Algorithm 4,

$$\text{Prob(output of IsMEMBERCONJUGATES is TRUE}|a^x \in L_i \setminus L_{i+1}) \leqslant e.$$

Thus the membership test condition holds for $(G_0, H, K, \text{IsMEMBERCONJUGATES})$ in G. $\square$

By Lemma 7.2, we can use IsMEMBERCONJUGATES(a^{xy}, e) to replace the algorithm IsMEMBER(xy, e) in the BASICSIFT Algorithm 2. Some explicit instances of IsMEMBER will be discussed in Section 10. We discuss here one special case, namely where $L_{i+1} = \langle a \rangle$. Here it turns out that Lemma 7.2 applies with $K = N_{G_0}(\langle a \rangle)$. Before proving this assertion in Lemma 7.3 below, we make a few comments about the context in which it will arise. (This context below occurs in several applications to sporadic simple groups.)

If condition (8) holds for a subgroup chain (5), then we construct, as at the beginning of this section, subsets $\mathcal{T}_i$ and $C_i = C_{G_0}(a)\mathcal{T}_i$, for each i, such that (6) and (7) both hold. Note that $a^{G_0} \cap L_i = a^{\mathcal{T}_i L_i}$, and that $\mathcal{T}_i L_i = \mathcal{T}_{i+1} L_i$ for each i; see Proposition 7.1. Also,

$\langle a \rangle \leqslant L_{k-1} \leqslant L_i$, for all $i \leqslant k-1$. This means that $a^{G_0} \cap L_i$ contains a, and hence contains a^{L_i}. Thus $\mathcal{T}_i$ contains an element of $C_{G_0}(a)L_i$. In particular, if $L_i \leqslant C_{G_0}(a)$, then $\mathcal{T}_i$ contains an element of $C_{G_0}(a)$. (Note, however, that this element of $\mathcal{T}_i$ need not be equal to 1.)

It is tempting to consider refining the chain (5) by inserting the subgroup $\langle a \rangle$ to obtain a new chain with its second-last subgroup equal to $\langle a \rangle$. However, condition (8) may fail to hold for this new chain. For example, if the original $L_{k-1} \cong \mathbb{Z}_2 \times \mathbb{Z}_2$, then a is an involution, and $|\mathcal{T}_{k-1}| = 3$, but only one of the three L_{k-1}-conjugacy classes in $a^{G_0} \cap L_{k-1}$ meets $\langle a \rangle$ non-trivially. Nevertheless, the situation $L_{k-1} = \langle a \rangle$ arises often in applications, so we end this section by extending the framework to include this case.

LEMMA 7.3. *Suppose that G, G_0, a, L_i, L_{i+1}, C_i, C_{i+1}, $\mathcal{T}_i$ and $\mathcal{T}_{i+1}$ are as in Proposition* 7.1, *that $H = C_i L_i$ and $K = C_{i+1} L_{i+1}$, and that $L_{i+1} = \langle a \rangle$. Then $K = C_{i+1} = N_{G_0}(\langle a \rangle)$ and*

$$|\mathcal{T}_{i+1}| = |N_{G_0}(\langle a \rangle) : C_{G_0}(a)| \leqslant \varphi(|a|).$$

Moreover, if $(G_0, L_i, \langle a \rangle, \text{IsMember})$ satisfies the membership test condition in G, for some algorithm IsMember, then so does $(G_0, H, N_{G_0}(\langle a \rangle), \text{IsMemberConjugates})$, where the algorithm IsMemberConjugates is given by Algorithm 4.

Proof. By the definition of $\mathcal{T}_{i+1}$ and L_{i+1}

$$a^{\mathcal{T}_{i+1} L_{i+1}} = a^{G_0} \cap L_{i+1} = a^{G_0} \cap \langle a \rangle = a^{N_{G_0}(\langle a \rangle)}.$$

However, L_{i+1} centralises $a^{N_{G_0}(\langle a \rangle)}$ and so $a^{\mathcal{T}_{i+1}} = a^{N_{G_0}(\langle a \rangle)}$, which implies that

$$C_{i+1} = C_{G_0}(a)\mathcal{T}_{i+1} = N_{G_0}(\langle a \rangle).$$

Moreover, since

$$L_{i+1} = \langle a \rangle \leqslant C_{G_0}(a) \subseteq N_{G_0}(\langle a \rangle) = C_{i+1},$$

we see that $K = C_{i+1}$.

Since L_{i+1} is abelian, $|\mathcal{T}_{i+1}| = |a^{G_0} \cap \langle a \rangle|$, and since $N_{G_0}(\langle a \rangle)$ acts on the set of $\varphi(|a|)$ generators of $\langle a \rangle$, with kernel $C_{G_0}(a)$ and with $a^{G_0} \cap L_{i+1}$ as one of the orbits, it follows that

$$|a^{G_0} \cap L_{i+1}| = |a^{N_{G_0}(\langle a \rangle)}| = |N_{G_0}(\langle a \rangle) : C_{G_0}(a)|.$$

The final assertion is part (b) of Lemma 7.2. □

8. IsMember *using element orders*

In this section we present a version of BasicSift that has proved useful, especially for the first link in a chain such as (1) for several sporadic simple groups G. It requires the relevant subsets to be subgroups. We give some applications that use this version in Section 10.

As in Section 7, we will describe a version of the procedure IsMember that can be used in the BasicSift Algorithms 2 and 3. Let G and G_0 be finite groups such that $G_0 \leqslant G$, and suppose that H and K are subgroups of G_0, with $K < H$. Therefore condition (3) automatically holds with $L = H$. An extra requirement is that for all subgroups M such that $K < M \leqslant H$, a reasonable proportion of the elements of M have orders that do not occur as orders of elements in K. We define

$$I = \{n \in \mathbb{N} \mid \text{some } M, \text{ where } K < M \leqslant H, \text{ has elements of order } n \text{ but } K \text{ does not}\}.$$

Algorithm 5: IsMemberOrders
```
/*See Proposition 8.1 for notation */
```

Input: (y, e) where $y \in G$, and $0 < e < 1/2$;
Output: True or False;
set $N = \lceil \log(e^{-1}) / \log((1 - p_0)^{-1}) \rceil$;
set $n = 0$;
repeat
    set $h = \text{RandomElement}(\langle K, y \rangle)$;
    if the order of h is in I **then**
        return False
    end
    set $n = n + 1$
until $n \geqslant N$;
return True

Algorithm 5: The algorithm IsMemberOrders.

Assume that $I \neq \varnothing$, and let p_0 be a number such that, for all M with $K < M \leqslant H$, the proportion of the elements of M with orders in I is at least p_0. We suppose that $p_0 > 0$. As usual, we assume that random selections in the procedure are made independently and uniformly from the relevant subgroups. Moreover, we emphasise that this is a 'black-box algorithm', and in particular it is not easy to find the order of an element efficiently. To test whether an element g has a particular order $n \in I$, we check first that $g^n = 1$, which implies that the order of g divides n, and then, for each maximal proper divisor d of n, we test that $g^d \neq 1$. We define $\bar{I}$ to be the number of integers that are either equal to, or a maximal proper divisor of, an element of I. Then for $g \in G_0$ we can test whether the order of g lies in I by examining $\bar{I}$ powers of g.

PROPOSITION 8.1. *Suppose that G, G_0, H, K, I, $\bar{I}$ and p_0 are as above. Also suppose that, for any M satisfying $K \leqslant M \leqslant H$, RandomElement$(M)$ returns uniformly distributed, independent random elements of M. Then $(G_0, H, K, \text{IsMember})$ satisfies the membership test condition in G, where IsMember is Algorithm 5. Further, the cost of running* IsMemberOrders$(\cdot, e)$ *is*

$$O\big(\log(e^{-1}) \cdot p_0^{-1} (\xi + \log(\max I) \cdot \bar{I} \cdot \varrho) \big),$$

where $\max I$ *is the maximum integer in I, and ϱ and ξ are upper bounds for the costs of a group operation in G, and making a random selection from any subgroup of the form $\langle K, g \rangle$ ($g \in G$), respectively.*

REMARK. In Algorithm 5 we have to make a random selection from a possibly different group $\langle K, y \rangle$ for every step of the loop. Because the known algorithms for producing (pseudo-)random elements in groups all involve an initialisation phase, the constant ξ here could be much bigger than the constant ρ or even the corresponding constant ξ in other algorithms of this paper.

Proof of Proposition 8.1. If $y \in K$, then by one of the conditions on the input, no element of $\langle K, y \rangle = K$ has order in I, and hence the output is True. Now suppose that $y \in H \setminus K$,

so that $K < \langle K, y \rangle \leqslant H$. By assumption, the proportion of elements of $\langle K, y \rangle$ with order in I is at least p_0. Thus, after N independent random selections from $\langle K, y \rangle$, the probability that we do not find at least one element with order in I is at most $(1 - p_0)^N$. The definition of N implies that $(1 - p_0)^N \leqslant e$. Thus the membership test condition is satisfied.

Now we estimate the cost. For each random $h \in \langle K, y \rangle$, we compute h^n for each n that is either equal to, or a maximal divisor of, an element of I. We do this by first computing $h^2, h^4, \ldots, h^{2^m}$, where $2^m \leqslant \max I < 2^{m+1}$. We use these elements to compute h^n, for each relevant n, with at most $m\bar{I}$ group multiplications. Thus the cost of computing all of the relevant h^n is at most $m\bar{I}\varrho = O(\log(\max I)\bar{I}\varrho)$. The number of random h to be processed is at most N, which, by Lemma 3.2, is $O(\log(e^{-1}) \cdot p_0^{-1})$. Thus an upper bound for the cost is $O(\log(e^{-1}) \cdot p_0^{-1} (\xi + \log(\max I) \cdot \bar{I} \cdot \varrho))$. □

In most cases when Algorithm 5 is used, we have K maximal in H, and so the only possibility for M in Proposition 8.1 is K or H. Also, it is often true that I consists entirely of primes, and then $\bar{I} = |I| + 1$.

COROLLARY 8.2. *Use the notation of Proposition 8.1 and suppose that $u = |H : K|$. Let BASICSIFT be Algorithm 2 with Algorithm 5 as ISMEMBER. Then the cost of executing BASICSIFT$(\cdot, \varepsilon)$ with $0 < \varepsilon < 1/2$ is*

$$O\left(\log(\varepsilon^{-1}) \cdot u \left(\xi + \varrho + \frac{\log(\varepsilon^{-1}) + \log u}{p_0} (\xi' + \log(\max I) \cdot \bar{I} \cdot \varrho)\right)\right),$$

where ξ is the cost of selecting a random element of H, ξ' is an upper bound for the cost of selecting a random element from a subgroup of the form $\langle K, x \rangle$, where $x \in H$, and ϱ is the cost of a group operation in G.

Proof. Using the notation of Theorem 5.3, since $H = L > K$ we have $p = |K|/|H|$, which is u^{-1}. Thus, by Theorem 5.3 and Proposition 8.1, the cost of this version of BASICSIFT$(\cdot, \varepsilon)$ is

$$O\left(\log(\varepsilon^{-1}) \cdot u(\xi + \varrho + \log(e^{-1}) p_0^{-1} (\xi' + \log(\max I) \cdot \bar{I} \cdot \varrho))\right),$$

where $e = \varepsilon u^{-1}/2(1 - u^{-1})$. Now

$$\log(e^{-1}) = \log(\varepsilon^{-1}) + \log(2) + \log(u - 1) = O(\log(\varepsilon^{-1}) + \log u),$$

and the assertion follows. □

9. *The Higman–Sims group* HS *revisited*

In Section 2 we presented a simple algorithm to write an element of HS as a word in a given generating set. This algorithm served as an example for the theory developed in this paper. We now examine how the steps of the HS algorithm in Section 2 fit into the theoretical framework presented in the subsequent sections. We use the notation of Section 2.

As in Section 2, G is a group isomorphic to HS, and we set $G_0 = G$. Let L_1 be a maximal subgroup of G isomorphic to $U_3(5).2$. Then L_1 has a subgroup Z of order 16. We noted in Section 2 that the proportion of elements of order 11 or 15 in HS is 41/165, while L_1 does not contain any such element. Let ISMEMBER$_1$ be Algorithm 5 with $I = \{11, 15\}$ and $p_0 = 41/165$. Then, by Proposition 8.1, $(G, G, L_1, $ISMEMBER$_1)$ satisfies the membership test condition in G. Let $C_1 = C_G(a)$, where $a \in Z$ and $|a| = 8$ as in Section 2, and let ISMEMBERCONJUGATES$_1$ be Algorithm 4 with ISMEMBER$_1$ as ISMEMBER.

Then, by Lemma 7.2, $(G, G, C_1L_1, \text{IsMemberConjugates}_1)$ also satisfies the membership test condition in G, and we use Algorithm 2 to obtain an algorithm BasicSift_1 such that $(G, G, C_1L_1, \text{BasicSift}_1)$ satisfies the basic sift condition in G.

In the next step we recall that $L_2 = 5^{1+2} : (8 : 2)$. We noted that $L_2 = N_G(Z(5^{1+2}))$, and so it is easy to design a deterministic algorithm IsMember_2 such that the 4-tuple $(G, L_1, L_2, \text{IsMember}_2)$ satisfies the membership test condition in G (just check whether a generator for $Z(5^{1+2})$ is mapped into $Z(5^{1+2})$). We set $C_2 = C_G(a)\mathcal{T}_2$ as in Section 2.2.

Using Algorithm 4, we find an algorithm $\text{IsMemberConjugates}_2$, using IsMember_2 as IsMember, such that $(G, C_1L_1, C_2L_2, \text{IsMemberConjugates}_2)$ also satisfies the membership test condition in G, and we use Algorithm 2 to build an algorithm BasicSift_2 so that $(G, C_1L_1, C_2L_2, \text{BasicSift}_2)$ satisfies the basic sift condition in G.

As L_3 is a cyclic group of order 8 and $C_3 = C_G(a)\mathcal{T}_3$ as in Section 2.3, it is easy to check membership in L_3, and following the procedure explained above, it is easy to obtain an algorithm BasicSift_3 such that $(G, C_2L_2, C_3L_3, \text{BasicSift}_3)$ satisfies the basic sift condition in G. In Section 2 we set $C_4 = C_G(a)$, and, using this fact, we can easily test membership in C_4. Thus the 4-tuple $(G, C_3L_3, C_4, \text{BasicSift}_4)$ can be constructed.

Finally, it is possible to list all 16 elements of C_4 and, via an exhaustive search, to construct an algorithm BasicSift_5 such that $(G, C_4, \{1\}, \text{BasicSift}_5)$ satisfies the basic sift condition in G.

Algorithm 1 can be used with

$$(G, G, C_1L_1, \text{BasicSift}_1),$$
$$(G, C_1L_1, C_2L_2, \text{BasicSift}_2),$$
$$(G, C_2L_2, C_3L_3, \text{BasicSift}_3),$$
$$(G, C_3L_3, C_4, \text{BasicSift}_4), \text{ and}$$
$$(G, C_4, \{1\}, \text{BasicSift}_5)$$

to sift an element through the chain

$$G \supset C_1L_1 \supset C_2L_2 \supset C_3L_3 \supset C_4 \supset \{1\}.$$

10. *Application of the results to sporadic simple groups*

An important part of the research presented here is to find explicitly a suitable subset chain (1) and a BasicSift algorithm for each step in this chain for many sporadic simple groups.

Note that all the example chains in this section provide pure black-box algorithms. No particular prior knowledge about the representations of the groups is used during the sifting. Of course, to construct the chains we made heavy use of lots of available information, and especially of 'nice' representations.

In the implementations, all the group elements that occur are expressed as straight-line programs in terms of standard generators in the sense of [17] and [18].

One could improve the performance by using specially crafted IsMember tests, relying on specific information about the given representation. Also, other methods will be better for certain representations.

In this section we assume that $G = G_0$ is one of the sporadic simple groups. For each group G, a subset S_i in the chain (1) will be a product $S_i = C_G(a)\mathcal{T}_iL_i$ with suitable a, $\mathcal{T}_i$ and L_i. We also set $C_i = C_G(a)\mathcal{T}_i$ and the sequence $C_1, \ldots, C_{k-1}$ will be referred to as

a *C-sequence*. The ingredients a, L_i and $\mathcal{T}_i$ are given in the tables in [Appendix A](#). In order to present the subset chains in the most compact form, we use the following notation.

The a-column. If the function IsMemberConjugates is used to sift through this step of the subset chain, then this column specifies the conjugacy class of a used by IsMemberConjugates. The conjugacy class is given using the Atlas notation; see [**7**]. We can assume without loss of generality that a is contained in all subgroups L_i where we need the hypothesis $a^G \cap L_i \neq \varnothing$. If the function IsMemberConjugates is not used in this step of the chain, then a dash is displayed in the appropriate cell.

The $C_G(a)$-column. This column contains information about the centralisers occurring in the C-sequence $C_1, \ldots, C_{k-1}$. Note that the C_i satisfy the conditions in (6).

The $|\mathcal{T}_i|$-column. Here we specify only the number of elements in $\mathcal{T}_i$. In each of the examples, we set $\mathcal{T}_0 = \{1\}$ and, for $i \geqslant 0$, the subset $\mathcal{T}_{i+1}$ is constructed using the procedure at the beginning of Section 7.

The L_i-column. In each table we list the subgroups $L_1, \ldots, L_{k-1}$ that are used to construct the subgroup chain (5); this chain will be referred to as 'the L-chain'. Each such subgroup is specified as precisely as necessary to define the descending subset chain. For example, in HS, the group L_1 is specified as $U_3(5).2$ (Atlas notation; see [**7**]), which means that any subgroup of G that is isomorphic to $U_3(5).2$ can play the rôle of L_1. Similarly, one may take L_2 to be any subgroup of L_1 that is the semidirect product of an extraspecial group of order 125 and a 2-group, as explained in the corresponding cell of the table.

The p-column. In this column we display the sifting parameter $p(C_{i-1}L_{i-1}, C_iL_i, L_{i-1})$ (see Definition 5.2 and Proposition 7.1).

The BasicSift-column (BS). We describe the BasicSift algorithm that is used in a particular step of the subset chain. The letter 'R' stands for BasicSiftRandom (see Algorithm 2) and the letter 'C' stands for BasicSiftCosetReps (see Algorithm 3). Note that in some cases Algorithm 3 is also used to try a certain set of group elements, such as the set $\mathcal{T}_i$ or its inverses.

The IsMember-column. In this column we describe how we test membership in the subgroup L_i. If an a is specified in the a-column, then we first design an algorithm IsMember for the pair (L_{i-1}, L_i) using the parameters in the same cell of the table. Then we use Algorithm 4 to obtain a new algorithm IsMember for the pair $(C_{i-1}L_{i-1}, C_iL_i)$, and finally, Algorithm 2 yields a 4-tuple $(G, C_{i-1}L_{i-1}, C_iL_i, \text{BasicSift}_i)$ satisfying the basic sift condition in G.

The membership test IsMember for the pair (L_{i-1}, L_i) is described using the following notation.

(a) If a set I of element orders is specified, Algorithm 5 is used for the IsMember test for L_i. In this case we also specify the probability p_0 to find an element of such an order in L_{i-1}.

(b) If, in the BASICSIFT-column of the table, an L_i is specified to be the centraliser or the normaliser of an element or a subgroup, then, using this fact, we build a deterministic algorithm to determine membership of L_i.

(c) Finally, the symbol 1 in that column indicates that we use an exhaustive search to test equality in the subgroup L_i. This method will be used in the special case when $L_i = 1$.

Note that the symbol '1' may stand either for the trivial subgroup or for the identity element, but its meaning is always clear from the context.

In practical implementations the sifting is carried out in several stages. In the first stage we sift our element into a smaller subgroup (usually a centraliser of an element), and then we start a new sifting procedure in that subgroup. We repeat this until we reach the trivial subgroup containing only the identity element. In the tables given in Appendix A, we indicate the boundary between different stages by a horizontal line. For instance in Table 1, we first sift our element into the subgroup 2.S4, and then carry out a new sifting procedure in $2.S_4$.

The first example is given in Table 1, which describes a subset chain for the sporadic simple Mathieu group M_{11}. In Table 2 we present another subset chain for M_{11} to demonstrate a new idea, namely that information gained during an ISMEMBER test can be used further. Table 3 contains a subset chain for the sporadic simple Mathieu group M_{12}. In Table 4 we describe a subset chain for the sporadic simple Mathieu group M_{22}. Table 5 presents a subset chain for the sporadic simple Janko group J_2, that uses only deterministic membership tests. In contrast, Table 6 shows another chain for J_2 with membership tests using element orders.

In Sections 2 and 9 we have already described the subgroup chain for the sporadic simple Higman–Sims group HS presented in Table 7. We found this chain very useful to illustrate the ideas used in this paper. However, it turns out that one can design a much more efficient chain for HS; the details are presented in Table 8.

We conclude this section with a larger example, in which we demonstrate yet another idea, namely that there may be 'branches' in chains, leading to different behaviour of the algorithm under certain circumstances that may occur during the calculation. See Table 9 for details, and Note (i) to Table 9 for an explanation.

We have implemented the generalised sifting algorithms using the subset chains described in the tables below for some of the sporadic simple groups. The implementations were written in the GAP 4 computational algebra system [8], and will be made available separately in the future. Information on the performance of our implementations can be found in Table 10, and in the notes to that table.

Appendix A.   *The tables*

Table 1: A chain for M_{11} using $2.S_4$.

M_{11}	a	$C_G(a)$	$|\mathcal{T}_i|$	L_i	p	BS	IsMember
1	$a \in 2A$	$2.S_4$	2	$2.S_4$	13/165	R	$C_G(a)$
2	$a \in 2A$	$2.S_4$	3	2^2	1/6	C	$C_{L_1}(b)$ with $b^2 = 1$
3	$a \in 2A$	$2.S_4$	1	1	1/3	C	1
3	–	–	–	$2.S_4$	–	–	–
4	–	–	1	8	1/6	C	$C_{L_3}(8A)$
5	–	–	1	1	1/8	C	1

Table 2: A second chain for M_{11} using $L_2(11)$.

M_{11}	a	$C_G(a)$	$|\mathcal{T}_i|$	L_i	p	BS	IsMember
1	$a \in 11A$	$\langle a \rangle$	1	$L_2(11)$	1/12	C	see notes below
2	$a \in 11A$	$\langle a \rangle$	1	$|N_G(\langle a \rangle)| = 55$	1/12	C	$N_G(\langle a \rangle)$
3	$a \in 11A$	$\langle a \rangle$	1	1	1/5	C	1
3	–	–	–	$\langle a \rangle$	–	–	–
4	–	–	1	1	1/11	C	1

Notes to Table 2.

Let a be as in the table, and select $x \in G$. We want to write the element x as a word in a given 'nice' generating set. Choose an element $a' \in 11A \cap L_1$ such that $[a, a'] \neq 1$, and let $z \in L_1$ with $(a')^z = a$. Then L_1 has 12 Sylow 11-subgroups, namely $\langle a \rangle$ and $\langle (a')^{a^i} \rangle$ for $i = 0, \dots, 10$. For $y_1 \in G$, $a^{xy_1} \in L_1$ if and only if $\langle a^{xy_1} \rangle$ coincides with one of the Sylow 11-subgroups of L_1. Further, such a Sylow subgroup is self-centralising in G. Thus the membership test $a^{xy_1} \in L_1$ is carried out by checking whether $[a^{xy_1}, a] = 1$ or $[a^{xy_1}, (a')^{a^i}] = 1$ for some $i \in \{0, \dots, 10\}$.

The second step of the sifting can be made more efficient as follows. Assume that $a^{xy_1} \in L_1$. If $[a^{xy_1}, a] = 1$, then $a^{xy_1} \in L_2$, and we can proceed to the third step of the sifting procedure. If $[a^{xy_1}, (a')^{a^i}] = 1$ then, for $y_2 = a^{11-i}z$ we have $a^{xy_1y_2} \in L_2$. Thus, storing some information about the membership test in the first step, we can immediately select the sifting element y_2 in the second step.

Table 3: A chain for M_{12}.

M_{12}	a	$C_G(a)$	$|\mathcal{T}_i|$	L_i	p	BS	IsMember
1	$a \in 2A$	$2 \times S_5$	1	$M_8 . S_4$	1/33	R	$C_G(2B)$
2	$a \in 2A$	$2 \times S_5$	1	$|C_{L_1}(x)| = 32$	1/3	C	$C_{L_1}(x)$ with $x^4 = 1$
3	$a \in 2A$	$2 \times S_5$	2	$|C_{L_1}(y)| = 8$	1/2	C	$C_{L_1}(y)$ with $y^4 = 1$
4	$a \in 2A$	$2 \times S_5$	1	1	1/2	C	$C_G(2A)$
4	–	–	–	$2 \times S_5$	–	–	–
5	–	–	1	$|N_{L_5}(z)| = 40$	1/6	C	$N_{L_5}(z)$ with $z^5 = 1$
6	–	–	1	$|C_{L_5}(z)| = 10$	1/4	C	$C_{L_5}(z)$ with $z^5 = 1$
7	–	–	1	1	1/10	C	1

Table 4: A chain for M_{22}.

M_{22}	a	$C_G(a)$	$|\mathcal{T}_i|$	L_i	p	BS	IsMember
1	$a \in 2A$	$2^4 : S_4$	1	$L_3(4)$	3/11	R	$I = \{6, 8, 11\}$, $p_0 = 103/364$
2	$a \in 2A$	$2^4 : S_4$	2	$2^4 : A_5$	5/21	R	$I = \{7\}$, $p_0 = 2/7$
3	$a \in 2A$	$2^4 : S_4$	1	1	1/60	R	$C_G(a)$ (see note below)
3	–	–	–	$2^4 : S_4$	–	–	–
4	–	–	1	2^4	1/24	C	$C_G(x)$ with $x^2 = 1$
5	–	–	1	1	1/16	C	1

Note to Table 4.

Here, the elements from $\mathcal{T}_2 = \{1, t_1\}$ are tried, together with elements from the group L_2, to reach the centraliser of a. The probability 1/60 is the minimum of the probability for the two cases $C_G(a) \cdot \{1\} \cdot L_2$ and $C_G(a) \cdot \{t_1\} \cdot L_2$.

Table 5: A chain for J_2 with deterministic membership tests.

J_2	a	$C_G(a)$	$|\mathcal{T}_i|$	L_i	p	BS	IsMember
1	$a \in 8A$	$\langle a \rangle$	2	$3.A_6.2_2$	$1/140$	R	$N_G(3A) = N_G(\mathrm{Soc}(L_1))$
2	$a \in 8A$	$\langle a \rangle$	4	$3^{1+2} : 8$	$1/5$	C	$N_G(3^{1+2}) = N_G(\mathrm{Syl}_3(L_1))$
3	$a \in 8A$	$\langle a \rangle$	4	8	$1/27$	C	$C_G(a) = \langle a \rangle$
4	$a \in 8A$	$\langle a \rangle$	1	1	$1/4$	C	$C_G(a)$
4	–	–	–	$\langle a \rangle$	–	–	–
5	–	–	1	1	$1/8$	C	1

Table 6: Another chain for J_2.

J_2	a	$C_G(a)$	$|\mathcal{T}_i|$	L_i	p	BS	IsMember	Note
1	$a \in 2A$	$2^{1+4}_- : A_5$	1	$3.A_6.2_2$	$1/6$	R	$N_G(3A)$	(i)
2	$a \in 2A$	$2^{1+4}_- : A_5$	1	$3 \times A_5$	$1/3$	C	$I = \{4, 12\}$, $p_0 = 1/4$	(ii)
3	$a \in 2A$	$2^{1+4}_- : A_5$	1	A_4	$1/5$	C	$I = \{5\}$, $p_0 = 2/5$	
4	$a \in 2A$	$2^{1+4}_- : A_5$	1	4	$1/3$	C	$C_{L_3}(a)$	
4	–	–	–	$2^{1+4}_- : A_5$	–	–	–	
5	–	–	1	$|L_5| = 192$	$1/10$	C	$C_{C_1}(2C)$	
6	–	–	1	$|L_6| = 32$	$1/6$	C	$N_{C_1}(4A)$	
7	–	–	1	$|L_7| = 16$	$1/2$	C	$C_{C_1}(4A)$	
8	–	–	1	1	$1/16$	C	1	

Notes to Table 6.

(i) $a^G \cap (3.A_6.2) \leqslant 3.A_6$, so we get an index 2 for free.

(ii) The 3 of $3 \times A_5$ is in $C_G(a)$, and hence $C_2 L_2 = 2 \times A_5$.

Table 7: The chain for HS from Sections 2 and 9.

HS	a	$C_G(a)$	$|\mathcal{T}_i|$	L_i	p	BS	IsMember
1	$a \in 8B$	2×8	1	$U_3(5).2$	$1/88$	R	$I = \{11, 15\}$, $p_0 = 41/165$
2	$a \in 8B$	2×8	2	$5^{1+2} : (8 : 2)$	$1/63$	R	$N_G(Z(5^{1+2}))$
3	$a \in 8B$	2×8	4	$\langle a \rangle$	$1/125$	R	$C_{L_1}(a)$
4	$a \in 8B$	2×8	1	1	$1/4$	C	1
4	–	–	–	2×8	–	–	–
5	–	–	1	1	$1/16$	C	1

Table 8: More efficient chain for HS.

HS	a	$C_G(a)$	$|\mathcal{T}_i|$	L_i	p	BS	IsMember	Note
1	$a \in 2A$	$4.2^4 : S_5$	1	M_{22}	$1/5$	R	$I = \{10, 12, 15, 20\}$, $p_0 = 7/20$	
2	$a \in 2A$	$4.2^4 : S_5$	1	$L_3(4)$	$3/11$	R	$I = \{6, 8, 11\}$, $p_0 = 103/264$	
3	$a \in 2A$	$4.2^4 : S_5$	1	A_6	$1/7$	R	$I = \{7\}$, $p_0 = 2/7$	
4	$a \in 2A$	$4.2^4 : S_5$	1	A_5	$1/3$	C	$I = \{4\}$, $p_0 = 1/4$	
5	$a \in 2A$	$4.2^4 : S_5$	1	A_4	$1/5$	C		(i)
6	$a \in 2A$	$4.2^4 : S_5$	1	2^2	$1/3$	C	$C := C_G(a)$	(ii)
6	–	–	–	$4.2^4 : S_5$	–	–	–	
7	–	–	1	$4.2^4 : A_5$	$1/2$	C	$C_C(4B)$	
8	–	–	1	$4.2^4.2^2$	$1/15$	C	$N_C(x^2)$ for some x with $x^8 = 1$	
9	–	–	1	$4.2^4.2$	$1/2$	C	$C_C(x^2)$	
10	–	–	1	8×2	$1/8$	C	$C_C(x)$	
11	–	–	1	1	$1/16$	C	1	

Notes to Table 8.

(i) The 2^2 in A_4 is equal to $C_{A_4}(a)$; therefore we can test membership of a^g in A_4 efficiently.

(ii) Here we reach $C_G(a)$, since $2^2 \leqslant C_G(a)$.

Table 9: A chain for Ly.

Ly	a	$C_G(a)$	$|\mathcal{T}_i|$	L_i	p	BS	IsMember	Note
1	$a \in 3A$	$3.\mathrm{McL}$	3	$3.\mathrm{McL}$	$15401/9606125$	R	$C_G(a)$	(i)
2	$a \in 3A$	$3.\mathrm{McL}$	1	$2.A_8$	$1/275$	R	$C_{L_1}(2A)$	(ii)
3	$a \in 3A$	$3.\mathrm{McL}$	3	$3 \times (2.A_5)$	$11/56$	R	$C_{L_2}(3A)$	(iii)
4	$a \in 3A$	$3.\mathrm{McL}$	3	$3 \times (2.S_3)$	$1/10$	C	$N_{L_3}(3B)$	
5	$a \in 3A$	$3.\mathrm{McL}$	4	$3 \times 2 \times 3$	$1/2$	C	set	(iv)
5	$-$	$-$	$-$	$3.\mathrm{McL}$	$-$	$-$	$-$	
6	$a' \in 3C$	$3^{2+4}.(2.A_5)$	1	$2.A_8$	$1/275$	R	$C_{3.\mathrm{McL}}(2A)$	(v)
7	$a' \in 3C$	$3^{2+4}.(2.A_5)$	3	$3 \times (2.A_5)$	$11/56$	R	$C_{L_6}(3A)$	
8	$a' \in 3C$	$3^{2+4}.(2.A_5)$	3	$3 \times (2.S_3)$	$1/10$	C	$N_{L_7}(3B)$	
9	$a' \in 3C$	$3^{2+4}.(2.A_5)$	4	$3 \times 2 \times 3$	$1/2$	C	set	(vi)
9	$-$	$-$	$-$	$3^{2+4}.(2.A_5)$	$-$	$-$	$-$	
10	$-$	$-$	1	$|L_{10}| = 1080$	$1/81$	C	$C_{C'}(z)$	(vii)
11	$-$	$-$	1	$|L_{11}| = 90$	$1/12$	C	$C_{C'}(z')$	(viii)
12	$-$	$-$	1	$|L_{12}| = 9$	$1/10$	C	$\mathrm{Syl}_3(L_{11})$	(ix)
13	$-$	$-$	1	1	$1/9$	C	1	

Notes to Table 9.

(i) $a^G \cap L_1 = \{a, a^{-1}\} \cup a^{xL_1}$ for some $x \in G$. We store an element $y \in G$ with $a^y = a^{-1}$, and handle the cases $a^g = a$ and $a^g = a^{-1}$ separately, which allows us to jump directly to step 6 in these cases. Otherwise, we can work with a single conjugacy class a^{xL_1} in L_1. Of course, most of the time this latter case will occur, as a^{xL_1} has 30800 elements.

(ii) The centraliser of a $2A$ element in $3.\mathrm{McL}$ is $3 \times (2.A_8)$. However, we have already avoided the 3 in the center by the special cases in step 1. Note that we have reduced the number $|\mathcal{T}_2|$ to 1, because $\mathcal{T}_2 = \{x\}$, again by the special cases in step 1.

(iii) L_3 is the centraliser in L_2 of an element of order 3.

(iv) In this step, we store the complete set of 4 possible results for a^g together with elements of G to conjugate them back to a. So we can reach $C_G(a)$ after this step with no additional costs.

(v) a' is obtained from $3C$ in $3.\mathrm{McL}$. By 3^{2+4} in L_6 we mean a 3-group with an elementary-abelian center of order 9 with an elementary-abelian group of order 81 as factor group. As in (ii), the centraliser of a $2A$ element in $3.\mathrm{McL}$ is $3 \times (2.A_8)$. However, since a' lies in $3C$ of $3.\mathrm{McL}$, we automatically reach $2.A_8$.

(vi) Note (iv) applies analogously.

(vii) z is an involution in $C' := C_{3.\mathrm{McL}}(a') = 3^{2+4}.(2.A_5)$.

(viii) z' is an element of order 15 in C'.

(ix) The Sylow-3-subgroup is normal; therefore just looking for element orders tests membership.

Table 10: Timings and number of multiplications for various chains.

Group	Time for 1000 calls (in seconds)	Average number of multiplications per call
$M_{11} \leqslant GL_{10}(2)$	0.8	116 (chain in Table 1)
$M_{11} \leqslant GL_{10}(2)$	0.7	187 (chain in Table 2)
$M_{11} \leqslant GL_{45}(3)$	26.1	116 (chain in Table 1)
$M_{11} \leqslant GL_{45}(3)$	34.1	185 (chain in Table 2)
$M_{12} \leqslant GL_{10}(2)$	2.2	243
$M_{12} \leqslant GL_{44}(2)$	11.6	245
$M_{12} \leqslant GL_{16}(11)$	10.9	238
$M_{22} \leqslant GL_{10}(2)$	25.2	1670
$M_{22} \leqslant GL_{21}(3)$	98.8	1685
$J_2 \leqslant GL_{36}(2)$	34.6	908 (chain in Table 5)
$J_2 \leqslant GL_{36}(2)$	46.2	847 (chain in Table 6)
$J_2 \leqslant GL_{14}(5)$	32.7	923 (chain in Table 5)
$J_2 \leqslant GL_{14}(5)$	33.2	846 (chain in Table 6)
$HS \leqslant GL_{20}(2)$	344.8	13923 (chain in Table 7)
$HS \leqslant GL_{20}(2)$	77.7	2783 (chain in Table 8)
$HS \leqslant GL_{49}(3)$	699.2	2807 (chain in Table 8)
$Ly \leqslant GL_{111}(5)$	19835.8	7416

Notes to Table 10.

The algorithms presented in this paper were implemented for the sporadic simple groups listed above. We used matrix representations of these groups, and Table 10 contains some average running times in seconds. For each representation, we sifted 1000 pseudo-random elements; the running times are for those 1000 calls to SIFT on a machine with a Pentium IV processor running at 2.53 GHz with 512 MB of main memory. The third column contains the average number of multiplications necessary for one call to SIFT, including the generation of pseudo-random elements. Note that the initialization phase of the pseudo-random generator (using product replacement) involves 100 multiplications for every newly generated group object. In all cases, the bound for the error probability was 1/100.

References

1. ROBERT M. BEALS, CHARLES R. LEEDHAM-GREEN, ALICE C. NIEMEYER, CHERYL E. PRAEGER and ÁKOS SERESS, 'Permutations with restricted cycle structure and an algorithmic application', *Combin. Probab. Comput.* 11 (2002) 447–464. 218, 219

2. ROBERT M. BEALS, CHARLES R. LEEDHAM-GREEN, ALICE C. NIEMEYER, CHERYL E. PRAEGER and ÁKOS SERESS, 'A black-box group algorithm for recognizing finite symmetric and alternating groups I', *Trans. Amer. Math. Soc.* 355 (2003) 2097–2113. 218, 219

3. WIEB BOSMA, JOHN CANNON and CATHERINE PLAYOUST, 'The Magma algebra system I: The user language', *J. Symbolic Comput.* 24 (1997) 235–265. 219

4. SERGEY BRATUS and IGOR PAK, 'Fast constructive recognition of a black box group isomorphic to S_n or A_n using Goldbach's conjecture', *J. Symbolic Comput.* 29 (2000) 33–57. 218

5. F. CELLER and C. R. LEEDHAM-GREEN, 'A constructive recognition algorithm for the special linear group', *The Atlas of finite groups: ten years on* (*Birmingham,* 1995) (Cambridge Univ. Press, Cambridge, 1998) 11–26. 218

6. GENE COOPERMAN, LARRY FINKELSTEIN and STEVE LINTON, 'Constructive recognition of a black box group isomorphic to $GL(n, 2)$', *Groups and computation II* (*New Brunswick, NJ,* 1995) (Amer. Math. Soc., Providence, RI, 1997) 85–100. 218

7. J. H. CONWAY, R. T. CURTIS, S. P. NORTON, R. A. PARKER and R. A. WILSON, *Atlas of finite groups* (Oxford University Press, Eynsham, 1985). 219, 241

8. THE GAP GROUP, 'GAP – groups, algorithms, and programming', Version 4.4, 2004, `http://www.gap-system.org`. 219, 242

9. P. E. HOLMES, S. A. LINTON, E. A. O'BRIEN, A. J. E. RYBA and R. A. WILSON, 'Constructive membership testing in black-box groups', preprint, Queen Mary, University of London, 2004. 218

10. WILLIAM M. KANTOR and ÁKOS SERESS, 'Black box classical groups', *Mem. Amer. Math. Soc.* 149 (708) (2001). 218

11. CHARLES R. LEEDHAM-GREEN, 'The computational matrix group project', *Groups and computation III*, OSU Math. Res. Inst. Publ. (ed. William M. Kantor and Ákos Seress, Walter de Gruyter, 2000) 85–101. 218

12. CHARLES R. LEEDHAM-GREEN, ALICE C. NIEMEYER, E.A. O'BRIEN and CHERYL E. PRAEGER, 'Recognising matrix groups over finite fields', *Computer algebra handbook. Foundations, applications, systems* (ed. V. Weispfenning, J. Grabmeier and E. Kaltofen, Springer, Berlin/New York, 2003) 459–460. 218

13. DEREK J. S. ROBINSON, *A course in the theory of groups* (Springer, New York/ Heidelberg/Berlin, 1982). 221

14. ÁKOS SERESS, *Permutation group algorithms* (Cambridge Univ. Press, 2003). 217

15. CHARLES C. SIMS, 'Computational methods in the study of permutation groups', *Computational problems in abstract algebra*, Proc. Conf. Oxford, 1967 (Pergamon Press, Oxford, 1970) 169–183. 217

16. CHARLES C. SIMS, 'Computing with subgroups of automorphism groups of finite groups', *Proceedings of the 1997 International Symposium on Symbolic and Algebraic Computation* (*Kihei, HI*) (ACM, New York, 1997) 400–403 (electronic). 217

17. R. A. WILSON, 'Standard generators for sporadic simple groups', *J. Algebra* 184 (1996) 505–515. 219, 240

18. R. A. WILSON, *et al.*, 'ATLAS of finite group representations', 2004, http://brauer.maths.qmul.ac.uk/Atlas/. 219, 240

Sophie Ambrose   sambrose@maths.uwa.edu.au
http://www.maths.uwa.edu.au/∼sambrose

School of Mathematics and Statistics
The University of Western Australia
35 Stirling Highway Crawley
Western Australia 6009
Australia

Max Neunhöffer   max.neunhoeffer@math.rwth-aachen.de
http://www.math.rwth-aachen.de/∼Max.Neunhoeffer

Lehrstuhl D für Mathematik
Rheinisch-Westfälische Technische Hochschule Aachen
Templergraben 64
52056 Aachen
Germany

Cheryl E. Praeger   praeger@maths.uwa.edu.au
http://www.maths.uwa.edu.au/∼praeger

School of Mathematics and Statistics
The University of Western Australia
35 Stirling Highway Crawley
Western Australia 6009
Australia

Csaba Schneider   csaba.schneider@sztaki.hu
http://www.sztaki.hu/∼schneider

Informatics Laboratory
Computer and Automation Research Institute
The Hungarian Academy of Sciences
1518 Budapest Pf. 63
Hungary