Chapter 14. Complexity of the Schreier-Sims Method

In this chapter we present some variations of the Schreier-Sims method together with analyses of their complexity. The final result is an $O(|\Omega|^5)$ bound.

Furst, Hopcroft, and Luks

The view taken by Furst, Hopcroft, and Luks is that the base of the group is [1,2,..|Ω|] and that the algorithm is constructing a table $T[1..|\Omega|, 1..|\Omega|]$ of permutations, where $T[i,j]$ is an element of the group G which fixes 1,2,..,i-1 and maps i to j. If no such element exists, then the entry is empty. Hence, each element along the diagonal can be taken to be the identity element. The non-empty elements of the i-th row form a set of coset representatives $U^{(i)}$ of $G^{(i+1)}$ in $G^{(i)}$.

During the algorithm the table data structure is just required to have $T[i,j]$ empty, or an element of G which fixes 1,2,...,i-1 and maps i to j. For such tables we can still define the set $U^{(i)}$ of non-empty entries of the i-th row. They define *closure* (T) to be the set of products $\{ u_r u_{r-1} \cdots u_1 \mid u_i \in U^{(i)} \}$, and define a table to be *complete* if its closure is G.

The *canonical representative* of an element $g \in G$ relative to a table T is the product $u_r u_{r-1} \cdots u_1$, where $u_i \in U^{(i)}$, such that $g = u_r u_{r-1} \cdots u_1$. Not all elements may have canonical representatives.

Lemma 1

A table T is complete if and only if both

(a) every generator of G has a canonical representative, and

(b) every product of (non-empty) entries of T has a canonical representative.

The procedure *sift* of Algorithm 1 will compute the canonical representative of an element if it exists. It is essentially the membership algorithm. If a canonical representative does not exist, then the table T is updated by replacing an empty entry by the residue of the stripping process.

Algorithm 1 : Sift

Input: a (not necessarily complete) table T for the group G;
an element $g \in G$;
Output: a modified table T such that $g \in closure(T)$;

```
procedure sift( g );
begin
  i := 1;
  while i ≤ |Ω| and T[i,i^g] is not empty do
    g := g × T[i, i^g]^-1;
    i := i+1;
  end while;
  if T[i,i^g] is empty then
    T[i,i^g] := g;
  end if;
end;
```

Lemma 2

Each invocation of *sift* is at worst $O(|\Omega|^2)$.

The algorithm then uses the characterization of Lemma 1 and repeated application of *sift* to force the table to be complete. One can take the non-empty entries of T to be a strong generating set for G relative to the given base.

Algorithm 2 : FHL Version of Schreier-Sims

```
Input: a set of generators of G;
Output: a complete table T for G;
begin
  initialize T so that the diagonal entries are the identity element
    and all other entries are empty;
  for each generator g of G do
    sift( g );
  end for;
  for each pair h, g of non-empty entries in T do
    sift( h × g );
  end for;
end.
```

Theorem 1

Algorithm 2 is $O(|\Omega|^6)$, provided that the size of the initial generating set is $O(|\Omega|^4)$.

Proof: There are at most $|\Omega|^2$ non-empty entries of T, so there are $O(|\Omega|^4)$ calls to sift. □

Consider the algorithm's execution for the automorphism group of the projective plane of order 2. The number of calls to *sift* is 443 (or 198 if one disregards pairs where one element is the identity). This is many more than the number of Schreier generators constructed by the Schreier-Sims method.

Knuth

Knuth uses a similar data structure to Furst, Hopcroft, and Luks but turns the algorithm on its head to get a nice recurrence relation. He takes the base to be [$|\Omega|$, $|\Omega|-1$, ..., 2, 1]. Then $G^{(i)}$ is the stabiliser of [$|\Omega|$, $|\Omega|-1$,...,$i+1$]. The entries of the table are such that $T[i,j]$ is an element of $G^{(i)}$ mapping i to j, or the entry is empty. Hence, the table is lower triangular with the identity element along the diagonal.

Define $T^{[j]}$ to be the upper left $j \times j$ subtable of T. All the elements in $T^{[j]}$ fix the points {$j+1$, $j+2$,..., $|\Omega|$ } and may be regarded as permutations of degree j.

Define $U^{(i)}$ to be the set of non-empty entries of the i-th row of T.

Define the *closure* of $T^{[j]}$ to be the set $\Gamma^{[j]}$ of products $u_1 u_2 \cdots u_j$, where $u_i \in U^{(i)}$.

Let S be a set of generators of G. (It will be extended to a strong generating set.) The data structure is the triple (S, T, Γ). The data structure is said to be *up to date of order m* if and only if $T^{[m]}$ is closed under multiplication.

If the table $T^{[m]}$ is closed under multiplication then $\Gamma^{[j]} = \langle S^{(j)} \rangle$, for all $j \leq m$. This implies that we can test membership in $\Gamma^{[j]}$.

Lemma 3

Membership testing in $\Gamma^{[j]}$ can be done in $O(j^2)$ steps.

The algorithm for the Schreier-Sims consists of a pair of mutually recursive routines A_j and B_j.

Algorithm 3 : Knuth Version of Schreier-Sims

procedure $A_j(g)$;

Input: (S, T, Γ) up to date of order j;
$g \in <S^{(j)}> - \Gamma^{[j]}$;
Output: (S, T, Γ) up to date of order j and $g \in S$;

begin
$S^{(j)} := S^{(j)} \cup \{g\}$;
for each $h \in U^{(j)}$ and $s \in S^{(j)}$ such that we do not know that $h \times s \in \Gamma^{[j]}$ **do**
$B_j(h \times s)$;
end for;
end;

procedure $B_j(g)$;

Input: (S, T, Γ) up to date of order $j-1$;
$g \in <S^{(j)}>$;
Output: (S, T, Γ) up to date of order $j-1$ and $g \in \Gamma^{[j]}$;

begin
$l := j^g$;
if $T[j,l]$ is empty **then**
$T[j,l] := g$;
else if $g \times T[j,l]^{-1} \notin \Gamma^{[j-1]}$ **then**
$A_{j-1}(g \times T[j,l]^{-1})$;

end if;
end;

Theorem 2

The algorithm is correct.

The proof shows that the set $\Gamma^{[j]}$ is closed under multiplication when A_j terminates. The proof uses induction on j and on the length of words in $S^{(j)}$.

The result is obviously true for $j = 1$, as $\Gamma^{[1]}$ is the identity subgroup.

Assume the result is true for $j-1$. Let $h_1, h_2 \in \Gamma^{[j]}$. We need to show that $h_1 \times h_2 \in \Gamma^{[j]}$. We can write h_1 as $h \times u_1$ for some $h \in \Gamma^{[j-1]}$, $u_1 \in U^{(j)}$; and we can write h_2 as $s_1 \times s_2 \times \cdots \times s_q$ for some $s_i \in S^{(j)}$. The call to B_j guarantees that $u_1 \times s_1 \in \Gamma^{[j]}$, so we can write $u_1 \times s_1$ as $g_1 \times u_2$ for some $g_1 \in \Gamma^{[j-1]}$, $u_2 \in U^{(j)}$. The call to B_j guarantees that $u_2 \times s_2 \in \Gamma^{[j]}$, so we can write $u_1 \times s_2$ as $g_2 \times u_3$ for some $g_2 \in \Gamma^{[j-1]}$, $u_3 \in U^{(j)}$. Continuing in this fashion we can write $h_1 \times h_2$ as a product $h \times g_1 \times g_2 \times \cdots \times g_q \times u_q$, where each $g_i \in \Gamma^{[j-1]}$, $h \in \Gamma^{[j-1]}$, and $u_q \in U^{(j)}$. So $h_1 \times h_2 \in \Gamma^{[j]}$. □

Lemma 4

In executing A_j, the number of calls to B_j is bounded by $|U^{(j)}|\times|S^{(j)}|$.

At this point, Knuth noticed that each call to A_j increases the size of the subgroup $\Gamma^{[j]}$, so the question was how many such calls could be made. This is equivalent to asking how long can a proper chain of subgroups be in the symmetric group of degree j. By Lagrange's Theorem, the indices in such a chain must be divisors of the order $j!$ of the symmetric group. Define $\Theta(n)$ to be the number of prime divisors of n, counting multiplicity. Then the number of calls to A_j is less than or equal to $\Theta(j!)$, which is known to be $O(j\text{loglog}(j))$.

Lemma 5

The cost of a sequence of calls to A_j is $O(j^5 \log\!log(j))$.

Proof: Let $C(j)$ be the total cost of calls to A_j. Then

$$C(j) = |U^{(j)}|\times|S^{(j)}|\times\left\lfloor j+j^2\right\rfloor + C(j-1)$$

where the first factor j is the cost of forming $h\times s$, and the second factor j^2 is the cost of testing membership using B_j. The last term $C(j-1)$ is the cost of calls to A_{j-1} from B_j. So

$$C(j) \le O(j^4\text{loglog}(j)) + C(j-1).$$

As $C(1) = 1$, we get that

$$C(j) \le \sum_{i=1}^{j} O(i^4\text{loglog}(i))$$

which is $O(j^5\text{loglog}(j))$.□

We must still insist that the number of generators supplied as input be of a reasonable size, that is $O(|\Omega|^3)$, as we must check whether each of them is in $\Gamma^{[|\Omega|]}$.

Theorem 3

The total cost of the algorithm is $O(|\Omega|^5 \log\!log(|\Omega|))$.

Tighter bounds on the length of proper chains of subgroups is the symmetric group were proven by Babai.

Theorem 4 (Babai)

The length of the longest proper chain of subgroups in the symmetric group of degree j is $O(j)$.

Corollary 1

Knuth's algorithm is $O(|\Omega|^5)$.

Jerrum

Jerrum assumes that the base is [1,2,..., $|\Omega|$] and introduced a new data structure for his view of the Schreier-Sims method. This data structure, a complete labelled branching, has also been used in later algorithms computing other group theoretic information. His algorithm is $O(|\Omega|^5)$ and uses $O(|\Omega|^2)$ storage, as opposed to the $O(|\Omega^3)$ storage requirements of Knuth's table.

A *branching* (Ω, E) is a acyclic directed graph (with no loops or multiple edges) with vertices Ω and edges E such that every vertex has at most edge directed into it.

A *labelled branching* is a triple (Ω, E, σ) such that

1. (Ω, E) is a branching;
2. $\sigma : E \rightarrow S_\Omega$ is a labelling of the edges by permutations;
3. for all edges ij, the label σ_{ij} stabilises 1,2,...,$i-1$; and
4. for all edges ij, the label σ_{ij} maps i to j.

For a directed path $P = i_0 i_1 i_2 .. i_t$, define the permutation σ_P to be the product

$$\sigma_{i_0 i_1} \times \sigma_{i_1 i_2} \times \cdots \times \sigma_{i_{t-1} i_t}$$

Hence, σ_P is an element mapping i_0 to i_t.

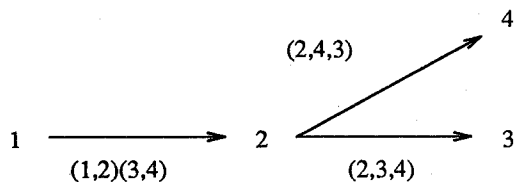
A *complete labelled branching* of G is a labelled branching where the edge labels are elements of G, and such that the set

$$U^{(i)} = \{ \sigma_P \mid P \text{ is a directed path with start vertex } i \}$$

is a right transversal of $G^{(i+1)}$ in $G^{(i)}$, for each i, $1 \leq i \leq |\Omega|$.

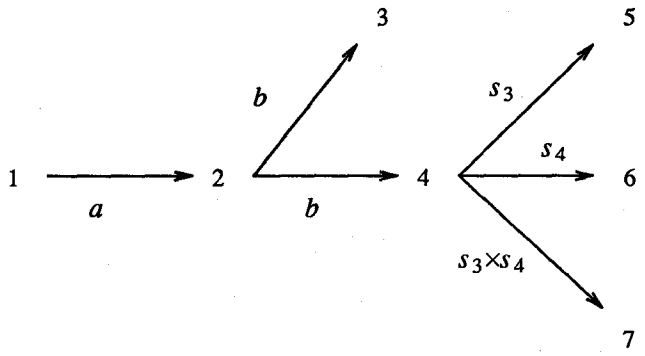
One can think of a labelled branching as containing the Schreier vectors for all levels of the stabiliser chain in a single data structure. Let us consider some examples of complete labelled branchings: one for the alternating group A_4, and one for the symmetries of the projective plane.

Figure 1: Complete labelled branching of A_4



There are at most $|\Omega|$ edges in a labelled branching, so the storage requirement is $O(|\Omega|^2)$. Of course, it is important that such a data structure exists for each group, so we will show how one can construct it from an existing base and strong generating set and Schreier vectors. Then we will show how to construct it from scratch; that is, from a generating set. That is Jerrum's version of the Schreier-Sims method.

Figure 2: Complete labelled branching of symmetries of the projective plane



Construction from table: Suppose we have constructed the table of Furst, Hopcroft, and Luks for a group G. Then we find the last non-empty entry $T[i,j]$ in the j-th column. For this entry we define an edge ij with label $T[i,j]$. This defines a complete labelled branching of G.

Construction from transversals $U^{(i)}$: Suppose G has a base $[1,2,...,k]$ with strong generators and basic transversals $U^{(i)}$, for $1 \le i \le k$. Then the following algorithm constructs a complete labelled branching.

Algorithm 4 : Complete labelled branching

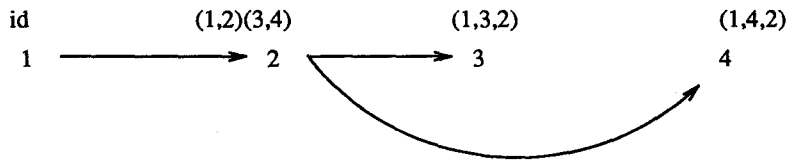
Input: a base $[1,2,...k]$ for G;
the basic transversals $U^{(i)}$ of G;
Output: a complete labelled branching of G;

begin
initialize the edge set E to be empty;
for $i := k$ to 1 **do**
for each u in $U^{(i)}$ **do**
if the vertex i^u has no incoming directed edge **then**
add an edge from i to i^u with label u to E;
end if;
end for;
end for;
end.

Construction by enumeration: Suppose we have a base $[1,2,...,k]$ for G with a corresponding strong generating set and Schreier vectors. Then we can enumerate the elements in lexicographic order. For each point j, we find the first element $g \in G$ which maps some point $i<j$ to j. (This will guarantee that i is as small as possible.) We add the edge ij and take σ_{ij} to be g.

There is an equivalent view of complete labelled branchings which labels vertices rather than edges. Define the vertex label τ_j to be σ_P, where P is the longest path which ends at j. Then σ_{ij} is $\tau_j \times \tau_i^{-1}$, and if P is the path $i_0 i_1 i_2 \cdots i_t$, then σ_P is $\tau_{i_t} \times \tau_{i_0}^{-1}$. Hence, one can readily convert from the edge label view to the vertex label view.

Figure 3: Complete (vertex) labelled branching of A_4



Note that if one uses the edge label view of the complete labelled branching then membership testing is $O(|\Omega|^3)$ as one must trace the Schreier vectors embedded in it. However, if one takes the vertex labelled view then one can calculate σ_P in $O(|\Omega|)$ steps, and hence membership testing is $O(|\Omega|^2)$.

Jerrum's algorithm uses the Schreier generators to determine the stabilizer chain, but can detect redundant Schreier generators at least good enough to reduce the number of generators from $O(|\Omega|^2)$ to less than $|\Omega|$. Hence, in his algorithm the number of generators for the stabilizer does not explode as he works down the chain.

Algorithm 5 : Jerrum Algorithm

Input: a set of generators for G;
Output: a complete labelled branching of G;
begin
  initialize the edge set E to be empty;
  for $i := 1$ **to** $|\Omega|$ **do**
    compute the orbit $\Delta^{(i)}$ and transversal $U^{(i)}$ from generators for $G^{(i)}$;
    augment E to include $U^{(i)}$;
    compute the $O(|\Omega|^2)$ Schreier generators of $G^{(i+1)}$;
    sift the Schreier generators to a set of generators of size less than $|\Omega|$;
  end for;
end.

procedure augment

Input: the edge set E and labelling;
    the level i;
    the basic orbit $\Delta^{(i)}$ and transversal $U^{(i)}$;
Output: an updated edge set E and labelling;

begin
  for each j in $\Delta^{(i)} - \{i\}$ **do**
    if edge lj exists in E **then**
      remove the edge lj;
    end if;
    add edge ij to E with label from $U^{(i)}$ mapping i to j;
  end for;
end;

We still need to explain the sifting process. This is presented in Algorithms 6 and 7.

Algorithm 6 : Sift a set of generators

procedure sift(**var** T : set of generators);

Input: a set T of generators for a group G;
Output: a set T of generators of G of size less than $|\Omega|$;

begin
  initialize Λ to be an empty labelled branching;
  for each $g \in T$ **do**
    sift(g);
  end for;
  T := labels of Λ;
end;

Algorithm 7 : Sift an element

procedure sift(g : element);

Input: a labelled branching Λ (with value Λ_0);
  an element g;
Output: an updated labelled branching Λ such that <labels(Λ)> = <labels(Λ_0), g>;

begin
  j := first point moved by g; $l := j^g$;
  while there exists a path from j to l in Λ **and** $j < |\Omega|$ **do**
    $g := g \times \tau_l^{-1} \times \tau_j$;
    j := first point moved by g; $l := j^g$;
  end while;
  if $j = |\Omega|$ **then**
    return;
  end if;
  while there exists an edge ml in Λ with $m > j$ **do**
    $g := g \times \sigma_{ml}^{-1}$; $l := m$;
  end while;
  if there exists an edge ml in Λ **then**
    add edge jl to Λ with label g;
    $g := \sigma_{ml} \times g^{-1}$;
    remove edge ml from Λ;
    update the vertex labelling;
    sift(g);
  else
    add edge jl to Λ with label g;
    update the vertex labelling;
  end if;
end;

The aim of Algorithm 7 is to express the element g as σ_P for some path P. The first part of the algorithm is just a variation of the membership test. During the second **while**-loop, one can think of the branching as being modified as in Figure 4, although this modification is not actually done. It modifies the branching as in Figure 5 in the **if**-clause at the end, or by simply adding an edge in the **else**-clause.

Figure 4

Figure 5

The vital ingredient to prove the correctness of the sifting process is that the group $\langle \text{labels}(B), g \rangle$ is invariant during *sift* (g). Once it is verified that each transformation of the labelled branching keeps the group invariant then it is clear the algorithm as a whole is correct.

Lemma 6

If the set T is $O(|\Omega|^2)$ then the cost of the call *sift* (T) is $O(|\Omega|^4)$.

If we ignore recursion, then the cost of each call *sift* (g) is $O(|\Omega|^2)$. However, each recursive call to *sift* occurs after

$$\text{edge length}(\Lambda) = \sum_{ij \in \Lambda} |j-i|$$

is reduced. The worst possible value of edge length is $|\Omega| \times (|\Omega|-1)/2$, so there are at most $O(|\Omega|^2)$ recursive calls to *sift* in total. □

Corollary 2

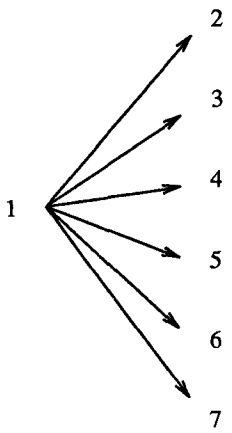
A complete labelled branching can be computed from a set of generators of size $O(|\Omega|)$ in time $O(|\Omega|^5)$.

Corollary 3

The group G has a strong generating set of size less than $|\Omega|$.

Let us consider the example of the symmetries of the projective plane. G is generated by a and b acting on $\{1,2,3,4,5,6,7\}$. For $i = 1$ we compute $U^{(1)} = \{ a^m \mid m = 0,1,...,6 \}$ and augment the empty labelled branching to obtain Figure 6 (omitting the labels).

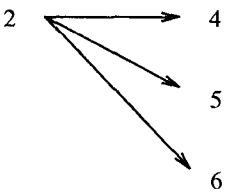
Figure 6



The non-trivial Schreier generators are b, (2,6,3,7)(4,5), (2,5,6)(3,4,7), (2,6,5)(3,7,4), (4,5)(6,7), (2,6)(3,7).

We call *sift* with this set and it processes each generator in turn. We start with an empty branching Λ. The sifting of the first three generators just adds an edge, so we have the branching in Figure 7.

Figure 7



The fourth generator is tripped by σ_{26} to give g=(4,6)(5,7). The branching is modified to that in Figure 8, and we recursively sift $\sigma_{26} \times g^{-1}$ = (2,4,7)(3,5,6). This is stripped by σ_{24} to (4,7)(5,6) and a new edge is added to give the branching in Figure 9.

Figure 8

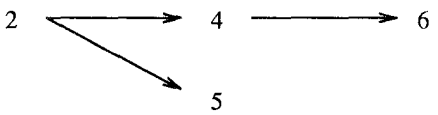
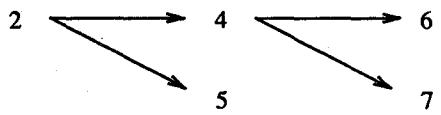
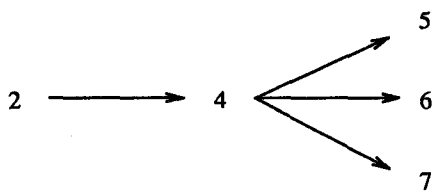


Figure 9



The fifth generator changes the branching to that in Figure 10, and we recursively sift $\sigma_{25} \times g^{-1} = (2,4,6)(3,5,7)$. This is stripped to the identity.

Figure 10

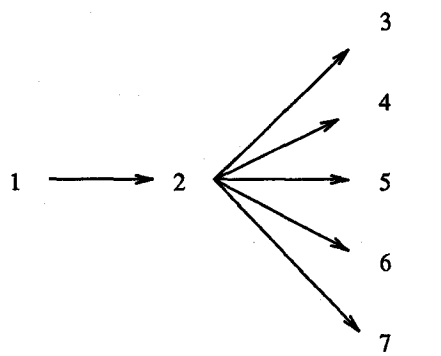


The sixth generator is stripped to the identity.

So we are finished. Our new generating set is
$\{ b, b \times (4,5)(6,7), b \times (4,6)(5,7), b \times (4,7)(5,6) \}$.

For $i = 2$, the Schreier vectors will give us $U^{(2)}$. The branching is augmented to that in Figure 11.

Figure 11



The non-trivial Schreier generators are (4,5)(6,7), (4,6)(5,7), (4,7)(5,6). We call *sift* with this set. It just builds up a labelled branching by adding an edge for each generator, so our resulting set of generators consists of the same three elements.

For $i = 3$, the set $U^{(3)}$ is constructed from the Schreier vectors. The branching is augmented to be that in Figure 2. The set of non-trivial Schreier generators is empty. Hence, we are finished. Our final strong generating set is
$\{$ (4,5)(6,7), (4,6)(5,7), (4,7)(5,6), (2,3)(6,7), b, a $\}$.
Note that the third and fourth are redundant.

Summary

We have presented some variations of the Schreier-Sims method which are simple enough to analyse. They all store actual permutations for the coset representatives. Jerrum's algorithm gives an $O(|\Omega|^5)$ time complexity and an $O(|\Omega|^2)$ space requirement.

Exercises

(1/Moderate) Execute Knuth's algorithm on the symmetries of the projective plane generated by a and b.

(2/Moderate) Which of the algorithms of this chapter and Algorithm 4 of the previous chapter would you expect to be fastest in practice? Why?

(3/Moderate) Repeat exercise 2 but modify Algorithm 4 of the previous chapter so it stores the coset representatives as permutations (and not in a Schreier vector).

(4/Difficult) Can the analyses of this chapter be carried over to Algorithm 4 of the previous chapter (the Schreier-Sims method)? Where does the difficulty lie?

Bibliographical Remarks

The first analysis of the Schreier-Sims method was presented by M. Furst, J. Hopcroft, and E. Luks, "*Polynomial-time algorithms for permutation groups*", (Proceedings of the IEEE 21st Annual Symposium on the Foundations of Computer Science, October 13-15, 1980), 36-41.

Knuth then presented his version of the Schreier-Sims method in an unpublished manuscript entitled "*Notes on efficient representation of perm groups*" in 1981. Jerrum's results were first presented in M. Jerrum, "*A compact representation for permutation groups*", (Proceedings of the IEEE 23rd Annual Symposium on the Foundations of Computer Science, 1982), 126-133, and later published in Journal of Algorithms **7** (1986) 60-78 in expanded form.

Since 1982 there has been progress in bounding the length of subgroup chains in the symmetric group by L. Babai, "*On the length of subgroup chains in the symmetric group*", Communications in Algebra **14** (1986) 1729-1736, who determined the $O(|\Omega|)$ bound on the length of chains, and minor progress in improving the complexity bounds of determining a base and strong generating set. L. Babai, E.M. Luks, and A. Seress, "*Fast management of permutation groups*", (Proceedings of the IEEE 29th Annual Symposium on the Foundations of Computer Science, October 24-26, 1988), 272-282, give an algorithm which is $O(|\Omega|^4 \log^c |\Omega|)$, where c is an undetermined constant. There has also been investigation of the parallel complexity of the problem by E.M. Luks and P. McKenzie, "*Fast parallel computation with permutation groups*", (Proceedings of the IEEE 26th Annual Symposium on the Foundations of Computer Science, 1985), 505-514, and L. Babai, E.M. Luks, and A. Seress, "*Permutation groups in NC*", (Proceedings of the 19th Annual ACM Symposium on Theory of Computing, May 25-27, 1987), 409-420.