Chapter 6. Lattice of Subgroups

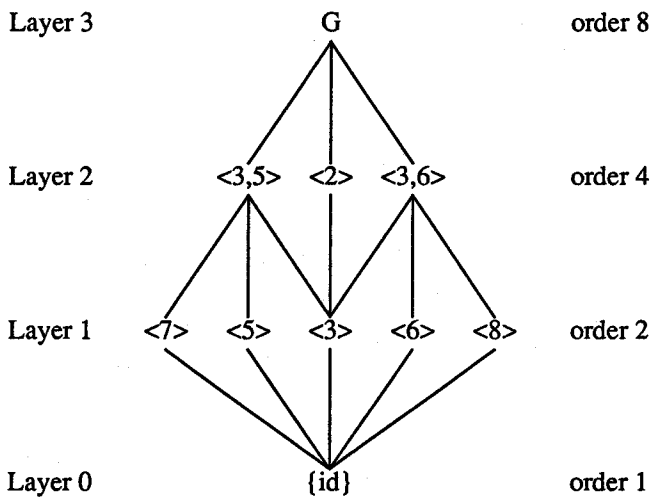
A very complete description of a group is a description of all its subgroups, and their relationship with one another. The subgroups of a group form a lattice. The lattice has layers allowing the subgroups to be generated layer by layer. This approach to their generation is called the cyclic extension method. This chapter presents the ideas behind the cyclic extension method, and then refines the ideas into an effective algorithm. An important space saving device is used to represent subgroups.

Unfortunately, the cyclic extension method cannot construct a certain type of subgroup. These are the perfect subgroups. We discuss briefly how the cyclic extension method could accommodate the perfect subgroups.

Lattices and Layers

A lattice is a set with a partial order, such as the inclusion operator $\leq$ of subgroups, where each pair of members has a least upper bound and a greatest lower bound. For the pair {H, K} of subgroups the least upper bound is the subgroup <H, K>, and the greatest lower bound is the intersection $H \cap K$. The lattice of subgroups of the symmetries of the square is shown in Figure 1.

Figure 1 : Subgroup Lattice of the Symmetries of the Square



The integers in Figure 1 represent the position of the elements in the original list of elements. The order of the subgroups is noted on the right. Note that the orders (from the top) are the products of three, two, one, and zero primes respectively (even though the primes are not distinct). Hence, we have layers three, two, one, and zero.

The members u of a finite lattice fall naturally into layers, depending on the longest chain of members

$$u_0 < u_1 < \cdots < u_i = u \tag{5.1}$$

that ends at u. Those with chains of length i belong to the i-th layer.

Certain subgroups U of a group have a chain of subgroups

$$id = U_0 < U_1 < \cdots < U_i = U$$

where U_{j-1} is normal in U_j, and the index $|U_j:U_{j-1}|$ is a prime p_j. These subgroups are called *soluble*, and they are amenable to generation by the cyclic extension method. A soluble subgroup in the i-th layer of the subgroup lattice will have an order that is the product of i (not necessarily distinct) primes. The index $|U:U_{i-1}|$ being prime means that the subgroup U can be generated by U_{i-1} and one extra element.

It is convenient to define the layers of the subgroup lattice in terms of the order of the subgroup rather than in terms of the chain (5.1). Thus, a subgroup U belongs to the i-th layer if the order of U is the product of i primes.

Constructing the Next Layer

This section looks at how the cyclic extension method constructs the subgroups in the i-th layer in the lattice from those in the (i-1)-st layer. The main problem is to avoid constructing the same subgroup many times over. First let us characterise the subgroups that do belong to the i-th layer.

A subgroup $U = U_i$ belongs to the i-th layer if there is a subgroup U_{i-1} in the (i-1)-st layer and an element g in the group such that

1. $U_i = < U_{i-1}, g >$,
2. U_{i-1} is normal in U_i, and
3. the index $|U_i:U_{i-1}| = p_i$, a prime.

For this we need an element g not in U_{i-1}, normalizing U_{i-1}, and with g^{p_i} in U_{i-1}, for some prime p_i.

Suppose we have constructed the (i-1)-st layer. To construct the i-th layer, we take each subgroup U_{i-1} in the (i-1)-st layer, find the elements g satisfying the above three conditions, and add $< U_{i-1}, g >$ to the i-th layer if it is not already in the i-th layer.

The obvious algorithm for the inductive step is Algorithm 1.

Algorithm 1 : Obvious Inductive Step

Input : a group G, given by a list of elements;
the $(i\text{-}1)$-st layer of subgroups of G;

Output : the i-th layer of subgroups of G;

begin
for each subgroup U_{i-1} in the $(i\text{-}1)$-st layer **do**
for each element g of G **do**
if $g \notin U_{i-1}$ **and** $g^{-1} \times U_{i-1} \times g = U_{i-1}$
and $g^p \in U_{i-1}$, for some prime p **then**
if $< U_{i-1}, g >$ is a new subgroup **then**
add $< U_{i-1}, g >$ to i-th layer;
end if;
end if;
end for;
end for;
end;

How to Test a Subgroup is New

For the subgroup $< U_{i-1}, g >$ to be a subgroup W already in the i-th layer, it is necessary and sufficient that $U_{i-1} \subseteq W$ and $g \in W$. If subgroups are represented by bitstrings, then these checks are simple bit operations. Furthermore, they can be performed without generating the subgroup $< U_{i-1}, g >$.

To form the subgroup $U = < U_{i-1}, g >$, we use the fact that U_{i-1} is normal in U and that $g^{p_i} \in U_{i-1}$. Hence, the coset decomposition of U is

$$U = U_{i-1} \cup \left[U_{i-1} \times g \right] \cup \left[U_{i-1} \times g^2 \right] \cup \cdots \cup \left[U_{i-1} \times g^{p_i - 1} \right].$$

So the full power of Dimino's algorithm is not required.

The number of tests to see if subgroups are new can be reduced by never testing elements that do not give a new subgroup. We reorganise the obvious algorithm to use a set Γ which contains the elements of the group that may generate a new subgroup containing U_{i-1}. We know

i. that $U_{i-1} \cap \Gamma$ is empty, and

ii. that $W \cap \Gamma$ is empty, for all subgroups W that contain U_{i-1} and are already in the i-th layer.

These facts lead to Algorithm 2.

Algorithm 2 : Inductive step avoidi

Input : a group G, given by a list of elements;
the $(i\text{-}1)$-st layer of subgroups of G;

Output : the i-th layer of subgroups of G;

begin

for each subgroup U_{i-1} in $(i\text{-}1)$-st layer **do**

(*initialise the set Γ to avoid old subgroups*)
$\Gamma := G - U_{i-1}$;
for each subgroup W already in i-th layer **do**
if $U_{i-1} \subseteq W$ **then** $\Gamma := \Gamma - W$; **end if**;
end for;

(*construct new subgroups*)
while Γ is not empty **do**

choose $g \in \Gamma$;
if $g^{-1} \times U_{i-1} \times g = U_{i-1}$ **and** $g^p \in U_{i-1}$, for some prime p **then**
add $U = < U_{i-1}, g >$ to i-th layer;
$\Gamma := \Gamma - U$;
else
$\Gamma := \Gamma - \{g\}$;
end if;

end while;

end for;

end;

Note that two of the conditions tested inside the inner loop of Algorithm 1 have been eliminated.

Eliminating the Normality Test

The normality test $g^{-1} \times U_{i-1} \times g = U_{i-1}$ can be eliminated by noting that the elements g which normalize U_{i-1} form a subgroup $N(U_{i-1})$. The search algorithms of Chapter 4 can compute the normalizer more efficiently than stepping through each of the elements of the group - which is basically what the inductive step is doing to find normalizing elements. The modified algorithm is Algorithm 3.

Algorithm 3 : Inductive step using normalizers

Input : a group G, given by a list of elements;
the (i-1)-st layer of subgroups of G;

Output : the i-th layer of subgroups of G;

begin

for each subgroup U_{i-1} in (i-1)-st layer **do**

(*initialise the set Γ to avoid old subgroups*)
(* and to consider only normalizing elements*)
compute $N(U_{i-1})$; $\Gamma := N(U_{i-1}) - U_{i-1}$;
for each subgroup W in i-th layer **do**
if $U_{i-1} \subseteq W$ **then** $\Gamma := \Gamma - W$; **end if**;
end for;

(*construct new subgroups*)
while Γ is not empty **do**

choose $g \in \Gamma$;
if $g^p \in U_{i-1}$, for some prime p **then**
add $U = < U_{i-1}, g >$ to i-th layer;
$\Gamma := \Gamma - U$;
else
$\Gamma := \Gamma - \{g\}$;
end if;

end while;

end for;

end;

While it is possible to discard more than just $\{g\}$ from Γ when $g^p \notin U_{i-1}$, for example the powers of g which have the same order as g, we will not follow that up. The next modification will take care of any such improvement.

Using Elements of Prime Power Order

This section shows that only the elements of G whose order is a prime power p^n, for some prime p and some exponent n, are required as extending elements. The following theorem justifies this claim.

Theorem

Suppose $U = < U_{i-1}, g >$ is in the i-th layer of the subgroup lattice, where U_{i-1} is in the (i-1)-st layer, and

1. U_{i-1} is normal in U, and
2. $g^p \in U_{i-1}$, for some prime p.

Let the order of g be $p^n \times r$, where p and r are coprime. Then g^r has order p^n, has its p-th power in U_{i-1}, and extends U_{i-1} to U.

The proof depends on r and p being coprime. This guarantees the existence of integers a and b such that

$$a \times p + b \times r = 1.$$

The fact that g^r has order p^n follows from $|g| = p^n r$. Since $g^p \in U_{i-1}$ and $(g^r)^p = (g^p)^r$, the p-th power of g^r is in U_{i-1}. The subgroup $< U_{i-1}, g^r >$ is U because g is in this subgroup. Indeed

$$g = g^{ap+br} = (g^p)^a \times (g^r)^b,$$

and $g^p \in U_{i-1}$. Thus completing the proof.

The effect of this result is that only elements g of prime power order are considered when extending the subgroups in layer i-1.

Using Subgroups of Prime Power Order

Amongst the elements of prime power order there is still a lot of redundancy. Suppose g has order p^n. Then the cyclic subgroup $< g >$ consists of the subgroup $< g^p >$ and the powers of g that have order p^n. The latter elements are g^r, where r and p are coprime. Since $<g> = <g^r>$, we have $<U_{i-1}, g> = <U_{i-1}, g^r>$, and so g will extend U_{i-1} to a subgroup U in the i-th layer if and only if g^r extends U_{i-1} to exactly the same subgroup U. Therefore it suffices to consider precisely one generator from each cyclic subgroup of prime power order.

Let Z be the set of cyclic subgroups of G of prime power order. For the symmetries of the square,

$$Z = \{ <2>, <3>, <5>, <6>, <7>, <8> \}.$$

Let *Z_generators* be a set of generators of the cyclic subgroups in Z. That is, choose one generator for each subgroup. For the symmetries of the square we have,

$$Z\_generators = \{ elt[2], elt[3], elt[5], elt[6], elt[7], elt[8] \}.$$

Typically *Z_generators* is much smaller than the whole group G.

The inductive step of the lattice algorithm only needs to scan the elements of *Z_generators* rather than G for elements satisfying the conditions. The algorithm is Algorithm 4.

Algorithm 4 : Inductive step using cyclic subgroup generators

Input : a group G, given by a list of elements;
the (i-1)-st layer of subgroups of G;
a set *Z_generators* of generators of the cyclic subgroups of G of prime power order;

Output : the i-th layer of subgroups of G;

begin

for each subgroup U_{i-1} in (i-1)-st layer **do**

(*initialise the set Γ to avoid old subgroups*)
(* and to consider only normalizing elements*)
(* and to use only the representative elements of prime power order *)
compute $N(U_{i-1})$; $\Gamma := Z\_generators \cap \left[N(U_{i-1}) - U_{i-1}\right]$;
for each subgroup W in i-th layer **do**
if $U_{i-1} \subseteq W$ **then** $\Gamma := \Gamma - W$; **end if**;
end for;

(*construct new subgroups*)
while Γ is not empty **do**
choose $g \in \Gamma$;
if $g^p \in U_{i-1}$, for some prime p **then**
add $U = < U_{i-1}, g >$ to i-th layer; $\Gamma := \Gamma - U$;
else
$\Gamma := \Gamma - \{g\}$; (*this is now the best possible*)
end if;
end while;

end for;

end;

There is only one prime relevant to each element in *Z_generators*, so the corresponding power g^p would already be known. Thus the remaining test in the inner loop is a simple bit operation.

The much smaller size of *Z_generators* than G makes our previous decision to compute normalizers by the search algorithm, rather than test that the candidates normalize U_{i-1}, suspect. The number of candidates is possibly quite small for any given subgroup U_{i-1}. Perhaps it is cheaper to test if they normalize U_{i-1} than to generate the whole normalizer. In practice, the normalizer is computed, but this decision is influenced by the fact that the normalizer of each subgroup is considered part of the description of the subgroup lattice. Thus it would be computed even if it were not used in the inductive step.

Full Lattice Algorithm for Soluble Subgroups

The first task of the full algorithm for computing the lattice of soluble subgroups is to calculate the cyclic subgroups of prime power order, and to choose a set of generators. This can be done by running through the elements of the group. The first layer will consist of the subgroups of order p, for some prime p. The cyclic subgroups of order p^n can be the first subgroups included in the n-th layer, if one so desires. If the order of the group is $\prod_{i=1}^{r} p_i^{n_i}$, then the last layer constructed is layer $(\sum_{i=1}^{r} n_i) - 1$. There is no need to construct the very top layer, since it consists of only G.

Analysis

This section analyses the full algorithm using each of the algorithms for the inductive step. Our major assumption is that bit operations are so cheap relative to element multiplications that they are performed at zero cost.

The cost of testing if an element g normalizes the subgroup U_{i-1} is $2 \times (i-1)$ multiplications to form the conjugates of the generators of U_{i-1}, and $(i-1)$ searches to find the position of the conjugates in the list of elements of G. This is equivalent to a total of $4 \times (i-1)$ multiplications.

The cost of testing $g^p \in U_{i-1}$ for all primes p involved in the order of g is small. We will approximate it by the constant c (about 8) multiplications per test.

The cost of forming a subgroup U as a bitstring is equivalent to $3 \times |U|$ multiplications, since we know the coset representatives of U_{i-1}. Although the subgroups involved in a layer will vary in size, this distinction will not be made in the formulae. Think of U_i as representing the "average" subgroup in layer i.

Let L_i be the number of subgroups in the i-th layer, and suppose that G belongs to the *top*-th layer. The total cost when using Algorithm 1 as the inductive step, and when all subgroups $< U_{i-1}, g >$ are formed is bounded by

$$\sum_{i=0}^{top-2} \left[L_i \times |G| \times \left[3 \times |U| + 4 \times i + c \right] \right]$$

multiplications because we extend layers 0 to *top*-2.

The total cost when using Algorithm 2 is bounded by

$$3 \times \sum_{U<G} |U| + \sum_{i=0}^{top-2} \left[L_i \times |G| \times \left[4 \times i + c \right] \right]$$

multiplications because now each subgroup is formed precisely once.

The total cost when using Algorithm 3 is bounded by

$$3 \times \sum_{U<G} |U| + \sum_{i=0}^{top-2} \left[L_i \times \left[cost\ of\ N(U_{i-1}) + c \times |N(U_{i-1})| \right] \right]$$

multiplications.

The cost of forming Z by determining the powers of each element is one multiplication and one comparison per element, because we deal with the powers of an element at the same time we deal with the element itself. The total cost when using algorithm 4 is bounded by

$$2 \times |G| + 3 \times \sum_{U<G} |U| + \sum_{i=0}^{top-2} \left[L_i \times cost\ of\ N(U_{i-1}) \right]$$

multiplications. If we run through *Z_generators* testing the normalizing condition rather than construct the normalizer then the cost is bounded by

$$2 \times |G| + 3 \times \sum_{U<G} |U| + 4 \times |Z| \times \sum_{i=0}^{top-2} \left[L_i \times i \right]$$

multiplications.

Saving Space

A group may have very many subgroups. So many in fact, that representing them as a characteristic function of the list of elements of the group will require a large amount of space. While such a characteristic function is needed to represent an arbitrary *subset*, an arbitrary *subgroup* can be represented more compactly as a characteristic function of the set Z of cyclic subgroups of prime power order. That is, a subgroup U is uniquely determined by the cyclic subgroups of prime power order that it contains. In fact, the elements of the subgroup are just the products of those in the cyclic subgroups of prime power order. If we list the subgroups of Z, or alternatively, list the elements of *Z_generators*, then a subgroup can be represented as a characteristic function of this list.

This still allows us to perform efficient bit operations to test inclusion of subgroups, form differences, and only slightly complicates forming $< U_{i-1}, g >$, since we must convert from the list of elements in the subgroup to the cyclic subgroups in the subgroup. This involves determining which elements of *Z_generators* are in the subgroup - these are simple bit operations.

In general, the size of Z is much smaller than the size of G, so this is a considerable saving in space.

A cyclic subgroup of prime power order is referred to as a *zuppo*, an acronym from the original German work on subgroup lattices.

Perfect Subgroups

The subgroups which cannot be constructed by the cyclic extension method are those with a chain

$$id \neq U_0 < U_1 < \cdots < U_i = U$$

(where U_{i-1} is normal of prime index in U_i) that cannot be prolonged as far down as the identity subgroup. The subgroup U_0 in these cases is a *perfect* subgroup. Once the perfect subgroups are included in the lattice, then the cyclic extension method will construct the other subgroups in the chain. So the problem is detecting and including the perfect subgroups.

All perfect subgroups of a group are contained in the unique largest perfect subgroup of the group. This is the last term in what is called the *derived series* of the group. The derived series can be constructed without difficulty, though we shall not discuss it here. There is a list of all perfect groups up to order 10 000 together with information on their structure, and the perfect subgroups they contain. The information is sufficient to easily distinguish any two perfect groups, and to reconcile the notation of the list with the actual description of the perfect group as a subgroup of G. Using this information, we can locate the last term of the derived series in the list, and include all its perfect subgroups into the lattice. The order of a perfect subgroup indicates where it should be added to the lattice of soluble subgroups.

Summary

With a knowledge of all the perfect groups, the cyclic extension method can determine all the subgroups of a group, and arrange them as a lattice. Clever use of bit operations and reducing the extending elements considered from the whole group to just the generators of the cyclic subgroups of prime power order saves a lot of time, and the use of characteristic functions of zuppos saves a lot of space.

Exercises

(1/Moderate) For the group in exercise 1 of Chapter 5, construct the subgroup lattice. It has no perfect subgroups.

(2/Moderate) For the symmetric group of degree 4 construct the subgroup lattice. It has no perfect subgroups.

(3/Moderate) Consider the following means of determining all the subgroups. Enumerate the $2^{|Z|}$ subsets S of *Z_generators*. For each subset S determine $<S>$ (as a characteristic function of the zuppos) using Dimino's algorithm. If $<S>$ is not already in the list of subgroups then add it to the list.

Execute this algorithm on the symmetries of the square.

(4/Moderate) Modify the algorithm of exercise 3 by keeping a set F of "filters". That is, if $<S> = G$ then add S to F. Before constructing $<S>$ test if there is some filter S' contained in S. If there is such a filter then we know that $<S> = G$.

Execute this algorithm on the symmetries of the square.

(5/Moderate) Order the enumeration of subsets S of *Z_generators* so that the subsets T containing S come in a consecutive sequence after S. Use this order of enumeration to improve the algorithm of exercise 4. That is, if we find a filter S' contained in S then the filter

is also contained in all subsets containing S. Hence, if the filters eliminate S we should automatically eliminate all subsets T containing S.

Execute this algorithm on the symmetries of the square.

(6/Moderate) The order of enumeration in exercise 5 allows us to improve the computation of $<S>$. Suppose $S = \{s_1, s_2, \ldots, s_m\}$. Then $S' = \{s_1, s_2, \ldots, s_{m-1}\}$ comes before S in the order of enumeration. If we have not eliminated S using the filters then S' was also not eliminated. Hence, $<S'>$ is some subgroup in the list of subgroups. Organize the algorithm to keep track of its location, and use the subgroup in constructing $<S>$.

The subgroup $<S'>$ can be used in two ways.

(i) if $s_m \in <S'>$ then there is no need to construct $<S>$ since this subgroup is $<S'>$.

(ii) if $s_m \notin <S'>$ then use the elements of $<S'>$ in the inductive step of Dimino's algorithm to compute $<S>$.

Modify the previous algorithm and execute it on the symmetries of the square.

(7/Moderate) A *maximal* subgroup H of G is a subgroup of G which has no larger subgroup U between H and G. That is, there is no subgroup U such that $H < U < G$.

Suppose that H is maximal and that $<S'> = H$. Then $S' \subseteq S$ implies that $<S>$ is H or G.

The set of filters F can be generalized to include subsets S' that generate G or a maximal subgroup (and this maximal subgroup has already been added to the list of subgroups).

How do we tell if $<S'>$ is maximal? In general we cannot (before we have constructed all the subgroups). However, there are some important special cases. If the index of H in G is a prime then H is maximal. So if we construct $<S'> = H$ with prime index then we can add S' to the set of filters.

Modify the algorithm of exercise 5 to incorporate this change and execute it on the symmetries of the square.

Bibliographical Remarks

The computation of subgroup lattices began in 1958 with the development and implementation of the cyclic extension method by Joachim Neubüser in Kiel. His work is published in J. Neubüser, "*Untersuchungen des Untergruppenverbandes endlicher Gruppen auf einer programmgesteuerten elektronischen Dualmaschine*", Numerische Mathematik **2** (1960) 280-292. The work has been continued by Neubüser and his students. The references are V. Felsch, **Ein Programm zur Berechnung der Untergruppenverbandes und der Automorphismengruppe einer endlichen Gruppe**, Diplomarbeit, Kiel, 1963; V. Felsch and J. Neubüser, *Ein Programm zur Berechnung des Untergruppenverbandes einer endlichen Gruppe*, Mitteilungen des Rheinisch-Westfälischen Institutes für Instrumentelle Mathematik (Bonn) **2** (1963) 39-74; P. Dreyer, **Ein Programm zur Berechnung der auflösbaren Untergruppen von Permutationsgruppen**, Diplomarbeit, Kiel, 1970; H. Steinmann, **Ein transportables Programm zur Bestimmung des Untergruppenverbandes von endlichen auflösbaren Gruppen**, Diplomarbeit, Aachen, 1974; Bernd Bohmann, **Ein Verfahren zur Bestimmung von Untergruppenverbänden, das die Benutzung von Sekundärspeicher zulässt, und seine Implementation**, Diplomarbeit, Aachen, 1978.

The program of Felsch (in 1963) contained information about the five smallest perfect groups. This was sufficient to complete the subgroup lattice of any group within the program's storage capability. The other programs mentioned above only determined the soluble subgroups. The list of perfect groups (up to order 10 000) was determined in G. Sandlöbes, "*Perfect groups of order less then* 10^4", Communications in Algebra, **9** (1981) 477-490 and used in the implementation described in V. Felsch and G. Sandlöbes, "*An interactive program for computing subgroups*", **Computational Group Theory**, (Proceedings of LMS Symposium on Computational Group Theory, Durham, 1982), edited by M. Atkinson, Academic Press, 1984, 137-143.

The construction of subgroups in layers really requires very few subgroups to actually be in memory. Indeed only information about the subset Γ, and the subgroups U and U_{i-1}, as well as information about the elements of G and the cyclic subgroups in Z are required. The subgroups of the (i-1)-st and i-th layer are processed sequentially, so their storage on disc is straightforward. In Neubüser's original implementation memory was synonymous with magnetic drums. Felsch in 1965 extended his own implementation to optionally use two magnetic tapes as external storage. Space was still restrictive as all the subgroups of a layer were required to fit into memory. While both Dreyer and Steinmann allowed the use of external storage, it was Bohmann who reorganised the cyclic extension method to no longer build the lattice layer by layer and thereby properly use external storage.

In practice, the lattice is augmented by information about normalizers and centralizers of subgroups, the conjugacy of subgroups, and other properties of subgroups. This represents a wealth of information about the group. Recent work has dealt with the problem of tailoring and interrogating this information, as it is too easy to be swamped by such a mass of detail. This work is described in the paper of V. Felsch and G. Sandlöbes.

There is an alternative to the cyclic extension method that was developed and implemented by L. Gerhards and W. Lindenberg, "*Ein Verfahren zur Berechnung des vollständigen Untergruppenverbandes endlicher Gruppen auf Dualmaschinen*", Numerische Mathematik **7** (1965) 1-10; and W. Lindenberg and L. Gerhards, "*Combinatorial construction by computer of a set of all subgroups of a finite group by composition of partial sets of its subgroups*", **Computational Problems in Abstract Algebra** (Proceedings of a conference, Oxford, 29 August - 2 September, 1967), J. Leech (editor), Pergamon Press, Oxford, 1970, 75-82. The main ideas of the method are presented in exercises 3-7. It is a combinatorial approach that has the advantage of constructing all the subgroups, including the perfect ones. As with the cyclic extension method, there is no adequate analysis of this method. Empirical timings indicate the method is competitive with the cyclic extension method for small groups with orders less than a few hundred. However, when the order of the group exceeds one thousand it suffers "combinatorial explosion" and is very slow. The cyclic extension method copes with groups with orders in the tens of thousands.

The determination of the derived series is described in J.J. Cannon, "*Computing local structure of large finite groups*", **Computers in Algebra and Number Theory** (Proceedings of a Symposium on Applied Mathematics, New York, 1970), G. Birkhoff and M. Hall, Jr (editors), SIAM-AMS Proceedings, volume **4**, American Mathematical Society, Providence, R.I., 1971, pp 161-176; and G. Butler and J.J. Cannon, "*Computing in permutation and matrix groups I: normal closure, commutator subgroups, series*", Mathematics of Computation, **39**,160 (Oct. 1982) 663-670.