J. Symbolic Computation (1998) **11**, 1–000

The groups of order at most 1000 except 512 and 768

Hans Ulrich Besche[†] and Bettina Eick[‡]

[†]*Lehrstuhl D für Mathematik, RWTH Aachen, 52056 Aachen, Germany*
[‡]*Mathematisches Institut, Universität Würzburg, 97074 Würzburg, Germany*

(Received 11 March 1997)

Recently, we developed practical algorithms to determine up to isomorphism the groups of a given order. Here we describe details on the implementations and the applications of these methods. In particular, we report on the determination of the groups of order at most 1000 except 512 and 768.

1. Introduction

In (Besche and Eick 1998) we introduce three methods to determine up to isomorphism the groups of a given order: the Frattini extension method to construct finite soluble groups, the cyclic split extension method to determine groups of order $p^n \cdot q$ with a normal Sylow subgroup for distinct primes p and q and the upwards extension method which is used to list insoluble groups.

Here, we first describe our implementations of these methods in Section 2. Since all three algorithms introduced in (Besche and Eick 1998) need a large machinery of group theoretic algorithms, it seems sensible to use one of the available systems for computational group theory as basis for the implementations; that is, GAP (Schönert *et al.* 1995) or MAGMA (Bosma *et al.* 1997). Both systems provide much of the underlying algorithms needed. We have chosen to use GAP for our purposes. The implementations will be available as GAP share package in GAP 4, see (The GAP Team 1998).

We used our implementations to determine up to isomorphism the groups of order at most 1000 except 512 and 768, see Section 3 for details. A table containing the numbers of groups of these orders is included there and a catalog with explicit descriptions of the determined groups is available in GAP 3.4.4 and MAGMA V2.4.

We mention that the groups of order 768 have been determined as well, see (Besche and Eick 1998a) and the groups of order 512 have been enumerated, see (Eick and O'Brien 1998). Considering the large number of groups of order 512 it seems not useful to determine these groups explicitly.

In Section 4 we give runtimes for some interesting applications of the methods and we report on the space requirements of our applications. Finally, in Section 5 we compare our results to others of similar type.

0747–7171/90/000000 + 00 $03.00/0

2. Implementations

GAP provides a large number of methods to compute with groups; for example, there are orbit-stabiliser methods, algorithms to compute conjugacy classes of elements or subgroups as well as various methods to compute automorphism groups and test isomorphism of groups. The practicality of all these methods depends on the structure of the input group as well as on its representation.

To faciliate effective computations with finite soluble groups in GAP it is most suitable to represent these groups by power commutator presentations. However, power commutator presentations are space consuming. Thus, during our computations, we represent as many power commutator presentations as possible by integers and use the presentations explicitly only if necessary. In Section 3.3 of (Besche and Eick 1998) there is a method described to convert a power commutator presentations to an integer and vice versa. To allow effective computations with insoluble groups we will mostly use permutation representations.

Furthermore, GAP provides a number of catalogs which have been useful for our applications. For example, E. A. O'Brien's catalog of groups of order dividing 2^8 and 3^6, M. Short's catalog of soluble irreducible subgroups of $GL(n,p)$ for $p^n < 256$ and D. Holt's and W. Plesken's catalog of perfect groups of order at most 10^4 (at most 10^6 with few exceptions) are available.

In the following sections we discuss the implementations of the three algorithms in more detail. We use the descriptions and the notation of Sections 3 to 6 of (Besche and Eick 1998) to which we refer by (B & E) for short.

2.1. The Frattini extension method

To compute candidates for the Frattini factors as described in Section 4.1 of (B & E), we need to determine subdirect products of irreducible soluble linear groups. For our purposes it almost always sufficed to use the GAP library of soluble irreducible linear groups. In a few cases we computed the irreducible representations over finite fields of groups of prime order; GAP provides a method to compute such modules up to equivalence. We implemented a method to compute subdirect products in GAP following the algorithm described in (Eick 1996).

The computation of minimal Frattini extensions, see Section 4.2 of (B & E), relies on methods to compute with extensions of finite soluble groups. The second cohomology group of a finite group and a module over the field $\mathbb{F}_p$ with p elements for a prime p is an elementary abelian p-group and hence can be represented by a vector space over $\mathbb{F}_p$. As part of the soluble quotient methods in GAP, there exists an algorithm to compute such an explicit description of the second cohomology group of a finite soluble group. Moreover, it is straightforward to obtain a power commutator presentation of an extension described in terms of a module and a 2-cocycle.

In Section 4.2.1 of (B & E) we outline a method to compute strong isomorphism classes of extensions of a finite soluble group. First we use the AutAg share package of GAP, see (Smith 1996), to compute the automorphism group of the group. Then we determine the group of compatible pairs by orbit-stabiliser calculations. Next the matrix action of the compatible pairs on the explicit description of the second cohomology group must be derived. This it technical, but elementary to implement. Then it remains to compute orbits of vectors under the action of a matrix group.

Finally, the Frattini extension method incorporates a method to reduce a list of finite soluble groups to isomorphism type representatives, see Section 4.3 of (B & E). The first step of this algorithm is to partition the given list of groups by invariants. In Section 4.3.1 of (B & E) there are suitable invariants described and their computation for a given group is elementary to implement in GAP. Next we have to reduce each sublist of groups by discarding isomorphic copies. Here we use the random method described in 4.3.2 of (B & E). This method relies on an algorithm to compute a random special polycyclic generating sequence for a finite soluble group which is elementary to implement using the random number generator of GAP and the GAP method to compute a (unique) special polycyclic generating sequence from an arbitrary polycyclic generating sequence. For each computed special polycyclic generating sequence we calculate the corresponding power commutator presentation and code it as an integer to save space and to provide an easy method to compare two power commutator presentations.

2.2. The cyclic split extensions method

This algorithm computes the groups of order $p^n \cdot q$ for distinct primes p and q with a normal Sylow subgroup, see Section 5 of (B & E). To apply it, we require descriptions of the groups of order p^n. First we have to compute the automorphism group for each such group and for this purpose we use a method implemented in GAP by B. Eick and E. A. O'Brien. This method will be available as a GAP share package.

For the next steps of the algorithm let G be a group of order p^n and let $A = Aut(G)$.

To compute the cyclic split extensions of $C_q \rtimes G$ we first compute all normal subgroups of G with cyclic factor of order dividing $q-1$. Then we compute orbits of these normal subgroups under action of A. Next for each orbit representative N we have to compute the stabiliser A_N of N in A and its induced subgroup U in $Aut(G/N)$. All of these steps are straightforward to implement. Now we need coset representatives of U in $Aut(G/N)$. This is straightforward, if we use a permutation representation of $Aut(G/N)$.

To compute the cyclic split extensions $G \rtimes C_q$ we first determine a permutation representation of A. Then we use the GAP permutation group method to compute 'rational classes of q-elements' to obtain conjugacy classes of subgroups of order q.

Moreover, it is straightforward to compute a power commutator presentation of the split extensions $C_q \rtimes G$ and $G \rtimes C_q$ which we then code as an integer to store it.

2.3. The upwards extension method

This method is used to construct insoluble groups and the most suitable representation to compute with such groups is as permutation groups. GAP provides a method to compute automorphism groups of such groups and test for isomorphisms. If the automorphism groups are represented as permutation groups as well, then the algorithm is elementary to implement in GAP.

The only non-trivial part is the construction of a permutation representation of a cyclic upwards extension G of a permutation group N where $[G : N] = p$. In (Huppert 1967), page 99, there is an explicit embedding of G in the wreath product of N by C_p outlined. Using this we can describe G as permutation group on $(d + 1) \cdot p$ points with $r + 1$ generators, where d is the degree and r is the number of generators of N.

For further computations with G it can be useful to try to reduce the degree and the generator number of G. For the first purpose we can try to determine an orbit or a block system of G with faithful G-action. For the second purpose we search for a small generating set of G using random elements of G.

3. Applications and results

We want to apply our methods to determine up to isomorphism the groups of order n. In general, this will be more difficult, if there exists a large number of such groups. This, in turn, depends mainly on the prime-power factorisation of n, say $n = p_1^{e_1} \cdots p_r^{e_r}$; for example, the number of groups of order n grows exponentially with increasing exponents e_i.

Our primary application of the methods described in Section 2 has been the determination of the groups of order at most 1000 except 512 and 768. Within this range there are a number of orders for which the corresponding determination of groups has been open; for example, the groups of order 192 had not been enumerated when we commenced on this project. Hence this range seems suitable to prove the practicality of our methods. However, our methods are not limited to this range and it will be possible to use these methods for larger orders with suitable prime-power factorisation as well.

To begin, note that the groups of prime-power order in our range are known. The 2- and 3-groups are available in the group library of GAP. For the groups of order 5^4 we used the p-group generation method included in the ANUPQ share package of GAP, see (O'Brien 1997), to compute them. All remaining p-groups have order at most p^3 and thus are abelian or extraspecial; hence they can be computed by the corresponding GAP functions.

Thus we consider the groups of orders with at least two prime divisors only. Then the interesting cases are the orders which factor into many primes. Clearly, the maximum number of prime factors of the orders in our range is 8 and the orders with 8 prime factors, i.e. 384, 576, 640, 864, 896 and 960, are our most difficult cases.

3.1. Construction of the groups of order at most 1000 except 512 and 768

First, the finite nilpotent groups can be constructed as direct products of p-groups. Hence the nilpotent groups are trivial to obtain using the available p-groups.

To determine the soluble, non-nilpotent groups in our range we first applied the Frattini extension method. This method was practical to list all such groups whose orders factors into at most 7 primes. Among the orders which factor into 8 primes the order 864 was easy and the orders 576 and 960 could be managed in a reasonable time, see Section 4. However, the three remaining orders were difficult.

Hence we introduced a special method for the orders 384, 640 and 896. All three orders are of the type $2^7 \cdot q$ and hence we can use the cyclic split extension method to compute all the non-nilpotent groups with a normal Sylow subgroup of these orders. Again, we rely on the GAP groups library of groups of order 2^7 here. The groups of this type included the most difficult cases for the Frattini extension method and it was practical to use the Frattini extension method for the remaining groups of these orders.

Finally, we computed the insoluble groups using the upwards extension method.

Altogether we obtained a catalog of groups which contains 174366 groups. The groups have been distributed as data libraries within GAP and MAGMA. In Table 1 and 2 we list the numbers of groups of these orders.

+	0	1	2	3	4	5	6	7	8	9
0		1	1	1	2	1	2	1	5	2
10	2	1	5	1	2	1	14	1	5	1
20	5	2	2	1	15	2	2	5	4	1
30	4	1	51	1	2	1	14	1	2	2
40	14	1	6	1	4	2	2	1	52	2
50	5	1	5	1	15	2	13	2	2	1
60	13	1	2	4	267	1	4	1	5	1
70	4	1	50	1	2	3	4	1	6	1
80	52	15	2	1	15	1	2	1	12	1
90	10	1	4	2	2	1	231	1	5	2
100	16	1	4	1	14	2	2	1	45	1
110	6	2	43	1	6	1	5	4	2	1
120	47	2	2	1	4	5	16	1	2328	2
130	4	1	10	1	2	5	15	1	4	1
140	11	1	2	1	197	1	2	6	5	1
150	13	1	12	2	4	2	18	1	2	1
160	238	1	55	1	5	2	2	1	57	2
170	4	5	4	1	4	2	42	1	2	1
180	37	1	4	2	12	1	6	1	4	13
190	4	1	1543	1	2	2	12	1	10	1
200	52	2	2	2	12	2	2	2	51	1
210	12	1	5	1	2	1	177	1	2	2
220	15	1	6	1	197	6	2	1	15	1
230	4	2	14	1	16	1	4	2	4	1
240	208	1	5	67	5	2	4	1	12	1
250	15	1	46	2	2	1	56092	1	6	1
260	15	2	2	1	39	1	4	1	4	1
270	30	1	54	5	2	4	10	1	2	4
280	40	1	4	1	4	2	4	1	1045	2
290	4	2	5	1	23	1	14	5	2	1
300	49	2	2	1	42	2	10	1	9	2
310	6	1	61	1	2	4	4	1	4	1
320	1640	1	4	1	176	2	2	2	15	1
330	12	1	4	5	2	1	228	1	5	1
340	15	1	18	5	12	1	2	1	12	1
350	10	14	195	1	4	2	5	2	2	1
360	162	2	2	3	11	1	6	1	42	2
370	4	1	15	1	4	7	12	1	60	1
380	11	2	2	1	20169	2	2	4	5	1
390	12	1	44	1	2	1	30	1	2	5
400	221	1	6	1	5	16	6	1	46	1
410	6	1	4	1	10	1	235	2	4	1
420	41	1	2	2	14	2	4	1	4	2
430	4	1	775	1	4	1	5	1	6	1
440	51	13	4	1	18	1	2	1	1396	1
450	34	1	5	2	2	1	54	1	2	5
460	11	1	12	1	51	4	2	1	55	1
470	4	2	12	1	6	2	11	2	2	1
480	1213	1	2	2	12	1	261	1	14	2
490	10	1	12	1	4	4	42	2	4	1
500	56	1	2	1	202	2	6	6	4	1

Figure 1. Numbers of groups of orders up to 509

+	0	1	2	3	4	5	6	7	8	9
510	8	1		15	2	1	15	1	4	1
520	49	1	10	1	4	6	2	1	170	2
530	4	2	9	1	4	1	12	1	2	2
540	119	1	2	2	246	1	24	1	5	4
550	16	1	39	1	2	2	4	1	16	1
560	180	1	2	1	10	1	2	49	12	1
570	12	1	11	1	4	2	8681	1	5	2
580	15	1	6	1	15	4	2	1	66	1
590	4	1	51	1	30	1	5	2	4	1
600	205	1	6	4	4	7	4	1	195	3
610	6	1	36	1	2	2	35	1	6	1
620	15	5	2	1	260	15	2	2	5	1
630	32	1	12	2	2	1	12	2	4	2
640	21541	1	4	1	9	2	4	1	757	1
650	10	5	4	1	6	2	53	5	4	1
660	40	1	2	2	12	1	18	1	4	2
670	4	1	1280	1	2	17	16	1	4	1
680	53	1	4	1	51	1	15	2	42	2
690	8	1	5	4	2	1	44	1	2	1
700	36	1	62	1	1387	1	2	1	10	1
710	6	4	15	1	12	2	4	1	2	1
720	840	1	5	2	5	2	13	1	40	504
730	4	1	18	1	2	6	195	2	10	1
740	15	5	4	1	54	1	2	2	11	1
750	39	1	42	1	4	2	189	1	2	2
760	39	1	6	1	4	2	2	1		1
770	12	1	5	1	16	4	15	5	2	1
780	53	1	4	5	172	1	4	1	5	1
790	4	2	137	1	2	1	4	1	24	1
800	1211	2	2	1	15	1	4	1	14	1
810	113	1	16	2	4	1	205	1	2	11
820	20	1	4	1	12	5	4	1	30	1
830	4	2	1630	2	6	1	9	13	2	1
840	186	2	2	1	4	2	10	2	51	2
850	10	1	10	1	4	5	12	1	12	1
860	11	2	2	1	4725	1	2	3	9	1
870	8	1	14	4	4	5	18	1	2	1
880	221	1	68	1	15	1	2	1	61	2
890	4	15	4	1	4	1	19349	2	2	1
900	150	1	4	7	15	2	6	1	4	2
910	8	1	222	1	2	4	5	1	30	1
920	39	2	2	1	34	2	2	4	235	1
930	18	2	5	1	2	2	222	1	4	2
940	11	1	6	1	42	13	4	1	15	1
950	10	1	42	1	10	2	4	1	2	1
960	11394	2	4	2	5	1	12	1	42	2
970	4	1	900	1	2	6	51	1	6	2
980	34	5	2	1	46	1	4	2	11	1
990	30	1	196	2	6	1	10	1	2	15
1000	199									

Figure 2. Numbers of groups of orders 510 up to 1000 except for 512 and 768

3.2. Remarks on the application of the Frattini extension method

Within the isomorphism test of the Frattini extension method there have been some choices left to the user, see Section 3.4 of (Besche and Eick 1998). We close this section with a description of the choices we used to compute the above catalog of groups.

As a first step in this algorithm, we compute invariants of the given list of groups based on partitions of conjugacy classes. At this stage of the algorithm we split the initial partition only by the first operation described in Section 3.4.1 of (B & E); that is, we distinguished classes which are fixed under 3rd, 5th or 7th powers from non-fixed ones. This yields an invariant which has been useful and is very efficient to compute even for large lists of groups.

Then the random isomorphism reduction is used. Here the user has to choose the parameter n which controls the probability of the random method. We have used 10 throughout our applications. With this choice, the random isomorphism test has been effective and, moreover, the random isomorphism test has always found all possible isomorphisms.

Finally, it must be verified that the random isomorphism reduction found all isomorphisms. Here we used as many of the remaining invariants based on partitions of conjugacy classes as necessary. In particular, splitting a partition using the fourth operation of "mapped words" together with the words $w(a,b) = [a,b] \cdot a^2$ and $w(a,b) = [a,b] \cdot a^3$ in the generators a and b has been very helpful. In a few cases these invariants did not suffice. Then we finished the verification using the general isomorphism test available in GAP .

4. Runtimes and space requirements

First we give some detailed runtimes for the Frattini extension method in Figure 3. We computed groups with given order and in some cases with prescribed properties. The table shows the time to compute the candidates for the Frattini factor groups and their number as well as the time used to extend them to groups of the desired order and their number. The rows in the table are sorted according to the number of Frattini extensions obtained. The runtimes were obtained using GAP under Linux on a 200 MHz Pentium Pro.

	Frattini free groups		Frattini extensions	
order (property)	number	time	number	time
$999 = 3^3 \cdot 37$	8	0:08	15	0:12
$640 = 2^7 \cdot 5$ (no normal Sylow subgroup)	5	0:37	21	2:51
$32 = 2^5$	5	0:02	51	1:55
$192 = 2^6 \cdot 3$ (no normal Sylow subgroup)	6	0:45	86	3:55
$720 = 2^4 \cdot 3^2 \cdot 5$ (non-supersoluble)	88	6:00	157	3:53
$816 = 2^4 \cdot 3 \cdot 17$ (non-nilpotent)	43	1:35	191	6:42
$864 = 2^5 \cdot 3^3$ (non-nilpotent)	210	9:24	4470	8:38:40
$960 = 2^6 \cdot 3 \cdot 5$ (non-nilpotent)	98	11:39	11020	32:14:47

Figure 3. Runtimes of the Frattini extension method as $(hours\ :)minutes\ :\ seconds$

Figure 3 contains some of the interesting cases in the construction of groups of order at most 1000, e.g. the groups of order 640 without normal Sylow subgroup and the non-nilpotent groups of orders 864 and 960. The timings show that the application of the Frattini extension method to such groups of order 640 has been very efficient. The groups of order 960 (and similarly the groups of order 576) have been by far the worst case in our applications of the Frattini extension method.

Secondly, we give an overview of the runtimes needed to set up our catalog of groups, see Figure 4. Again, the runtimes have been obtained using GAP under Linux on a 200 MHz Pentium Pro.

In the application of the cyclic split extension method we had to extend 2328 groups of order 2^7 by cyclic groups of order 3, 5 and 7. As the table shows, this method has proved to be extremely effective, since it needed a comparatively short time to determine roughly a third of all groups in the catalog. Of course, this is in part because we rely on the catalog of groups of order 2^7.

Using the upwards extension method we considered the 10 perfect groups of order at most 1000 and we had to extend the 5 groups which have order less than 500. The major part of the runtime has been used to compute automorphism groups and test isomorphism of groups. Here the worst case has been the determination of the 107 insoluble groups of order 960.

method	number of groups	time
Frattini extension	48510	121:38
cyclic split extension	53517	3:18
upwards extension	208	20:30
nilpotent	72131	–
Σ	174366	

Figure 4. Runtimes for the determination of the catalog as $hours : minutes$

Finally, we report on the space requirements of our implementations in the computation of the group catalog. Since almost all soluble groups are stored as an integer at any stage of the computations the space requirements of the Frattini extension method and the cyclic split extension method are moderate. The upwards extension method was used to compute few groups only and hence this computation was also not space critical. Thus all our computations are possible within 20 megabytes of workspace.

5. Accuracy of results

Our catalog has been computed without any human interaction. There is a risk of mistakes in our implementations or even in the underlying system GAP. However, we consider this risk in results obtained by computation smaller than by hand calculation, since at least the risk of accidental omissions of cases or other “local” errors is avoided.

To reduce the chance of mistakes in the implementations we checked our results. It is comparatively easy to justify that the groups in our lists are not isomorphic using different methods to determine isomorphism. The critical point is to make sure that we obtained all isomorphism types of groups.

First we compared the numbers of groups in our catalog with other sources; that is, with the groups of order up to 100 in the "SolvableGroup" library of GAP, with the catalog of soluble groups of order up to 242 computed by A. Betten, see (Betten 1998) and with the numbers of groups of orders 101 – 215 omitting 128 and 192 determined by J. K. Senior and A. C. Lunn, see (Senior and Lunn 1934) and (Senior and Lunn 1935). All numbers agreed with each other and, in particular, with our catalog. Thus the risk of mistakes among the groups of order up to 242 is very small.

Then we compared our catalog to the library of transitive groups of degree at most 22 as determined by A. Hulpke, see (Hulpke 1996). Here we checked that each isomorphism type of the soluble transitive groups having orders in the given range occurs in our catalog.

We once again constructed certain groups using other methods. For example, we computed all direct products of groups in the catalog and identified their isomorphism type. Moreover, we used the cyclic split extension method on various orders other than $2^7 \cdot q$ and checked that we find all isomorphism types of computed groups in our catalog.

References

Besche, H. U., Eick, B. (1998). Construction of finite groups. *J. Symb. Comp.*, this volume.

Besche, H. U., Eick, B. (1998a). The groups of order $2^n \cdot p$ for $n \leq 8$ and p an odd prime. In preparation.

Betten, A. (1998). Parallel construction of finite soluble groups. In preparation.

Bosma, W., Cannon, J., Playoust, C. (1997). The magma algebra system I: The user language. *J. Symb. Comp.*, 24, 3/4:235 – 265.

Eick, B. (1996). *Charakterisierung und Konstruktion von Frattinigruppen und Anwendungen in der Konstruktion endlicher Gruppen*. PhD thesis, RWTH Aachen. Also available as: Aachener Beiträge zur Mathematik 17.

Eick, B., O'Brien, E. A. (1998). Enumerating p-groups. In preparation.

Hulpke, A. (1996). *Konstrukton transitiver Pemutationsgruppen*. PhD thesis, RWTH Aachen. Also available as: Aachener Beiträge zur Mathematik 18.

Huppert, B. (1967). *Endliche Gruppen I*. Grundlehren der mathematischen Wissenschaften 134. Springer Heidelberg.

O'Brien, E. A. (1997). ANUPQ – Version 1.4. Share package of GAP 3. See (Schönert *et al.* 1995).

Schönert, M. *et al.*(1995). *GAP - Groups, Algorithms and Programming*, 5th edition.

Senior, J. K., Lunn, A. C. (1934). Determination of the groups of orders 101 – 161, omitting order 128. *Amer. J. Math.*, 56:328 – 338.

Senior, J. K., Lunn, A. C. (1935). Determination of the groups of orders 162 – 215, omitting order 192. *Amer. J. Math.*, 57:254 – 260.

Smith, M. J. (1996). AutAg. Share package of GAP 3. See (Schönert *et al.* 1995).

The GAP Team (1998). *GAP – Groups, Algorithms and Programming, Version 4.*