MATHEMATICS OF COMPUTATION
VOLUME 63, NUMBER 207
JULY 1994, PAGES 351-359

COMPUTING IRREDUCIBLE REPRESENTATIONS OF SUPERSOLVABLE GROUPS

ULRICH BAUM AND MICHAEL CLAUSEN

ABSTRACT. Recently, it has been shown that the ordinary irreducible representations of a supersolvable group G of order n given by a power-commutator presentation can be constructed in time $O(n^2 \log n)$. We present an improved algorithm with running time $O(n \log n)$.

1. INTRODUCTION

In general, computing the ordinary irreducible representations of a given finite group seems to be a hard problem. In their 1990 breakthrough paper [1], Babai and Rónyai have shown that this problem can be solved in time polynomial in the order of the group. Yet their (general-purpose) algorithm does not appear feasible in a practical sense.

However, one can do much better for special classes of finite groups. In this paper, we consider the class of supersolvable groups. Recall that these are finite groups whose chief factors are cyclic of prime order. Recently, Baum [2] has shown that the irreducible representations of supersolvable groups G, given by a power-commutator presentation, can be constructed in time $O(|G|^2 \log |G|)$. This algorithm has been implemented and has proved quite efficient in practice. Its main tool is the concept of symmetry-adapted representations, which will be briefly discussed in the next section. It has been shown in [2] that the irreducible representations of a supersolvable group adapted to a chief series are always monomial. The proof gives rise to an efficient construction along the chief series of the irreducible representations over any field containing a primitive eth root of unity, where e is the exponent of the group. During the construction, no field arithmetic is needed at all. Only symbolic calculations in the group of eth roots of unity are required. Hence, the symbolic result is valid if interpreted over any field containing a primitive eth root of unity. The structure of this algorithm is basically bottom-up. However, it contains a recursive subroutine, which turned out to be the most expensive part.

In this paper, we present an improved algorithm whose running time has order $|G| \log |G|$. It is based on the same ideas as the old algorithm, but the procedure has been completely reorganized to be purely bottom-up and nonrecursive.

Received by the editor October 13, 1992.
1991 *Mathematics Subject Classification*. Primary 20C15, 20C40; Secondary 68Q40.

2. Symmetry-adapted representations

In this section, we recall the concept of symmetry-adapted representations, which will be an important tool in our construction. Let G be a finite group of exponent e, and K a field containing a primitive eth root of unity. (According to a theorem by R. Brauer, K is a splitting field for every subgroup of G.) Let $\mathcal{T} = (G = G_n > \cdots > G_0 = \{1\})$ be a chain of subgroups of G. A matrix representation D of G over K is called *$\mathcal{T}$-adapted* if for all j, $0 \leq j \leq n$, the following two conditions hold:

- the restriction $D \downarrow G_j$ of D to G_j is *equal* to a direct sum of irreducible matrix representations of G_j;
- equivalent irreducible constituents of $D \downarrow G_j$ are *equal.*

Symmetry-adapted representations have been applied to various mathematical and physical problems, see, e.g., [4]. In particular, the most efficient algorithms for computing discrete Fourier transforms on finite groups are based on this concept [2, 3].

It is easy to see that every representation of G is equivalent to a $\mathcal{T}$-adapted representation. In our construction, $\mathcal{T}$ will be a chief series of the supersolvable group G; i.e., every G_i is normal in G and all indices $[G_i : G_{i-1}] =: p_i$ are prime. In this case, $\mathcal{T}$-adapted representations are almost unique.

Theorem 2.1. *Let $\mathcal{T}$ be a chief series of the supersolvable group G, and suppose that D and Δ are two equivalent irreducible and $\mathcal{T}$-adapted K-representations of G of degree d. Then the intertwining space*

$$\mathrm{Int}(D, \Delta) = \{X \in K^{d \times d} | XD(g) = \Delta(g)X \textit{ for all } g \in G\}$$

contains a monomial *matrix.*

A proof can be found in [2]. This result will be the basis of our construction described in the next section. In fact, we will see that the irreducible representations of a supersolvable group adapted to a chief series are themselves monomial; i.e., the representing matrices of all group elements are monomial.

3. The algorithm

Let $\mathcal{T} = (G = G_n > G_{n-1} > \cdots > G_0 = \{1\})$ be a chief series of the supersolvable group G, and fix generators $g_1, \ldots, g_n$ of G with $g_i \in G_i \backslash G_{i-1}$. For $1 \leq i \leq n$, define $\mathcal{T}_i = (G_i > G_{i-1} > \cdots > G_0)$. Let K be a field containing a primitive eth root of unity, where e is the exponent of G. We will see that our construction only involves (symbolic) calculations in the group of eth roots of unity; no general K-arithmetic is needed. To this end, we call a monomial matrix *e-monomial* if all of its nonzero entries are eth roots of unity. A representation is called e-monomial if the representing matrices of all group elements are e-monomial. Recall that for a representation F of a normal subgroup $N \lhd G$ and an element $g \in G$, the *conjugate representation* F^g of N is defined by $N \ni n \mapsto F^g(n) := F(g^{-1}ng)$.

Our algorithm works bottom-up along $\mathcal{T}$. At level i, $1 \leq i \leq n$, it takes the following input:

(1) $\mathcal{F}$, a full set of nonequivalent irreducible e-monomial representations of G_{i-1} over K such that $\bigoplus_{F \in \mathcal{F}} F$ is $\mathcal{T}_{i-1}$-adapted;

(2) For every $i-1 < j \leq n$, a permutation π_j of $\mathcal{F}$ such that $F^{g_j} \sim \pi_j F$ for all $F \in \mathcal{F}$ as well as e-monomial matrices $X_{jF} \in \operatorname{Int}(F^{g_j}, \pi_j F)$, $F \in \mathcal{F}$

and computes the following output:

(1) $\mathcal{D}$, a full set of nonequivalent irreducible e-monomial representations of G_i over K such that $\bigoplus_{D\in\mathcal{D}} D$ is $\mathcal{T}_i$-adapted;

(2) For every $i < j \leq n$, a permutation τ_j of $\mathcal{D}$ such that $D^{g_j} \sim \tau_j D$ for all $D \in \mathcal{D}$ as well as e-monomial matrices $Y_{jD} \in \operatorname{Int}(D^{g_j}, \tau_j D)$, $D \in \mathcal{D}$.

Note that the input of level 1 is trivial. Level i of the algorithm consists of two phases.

Phase 1. Computation of $\mathcal{D}$. Consider $F \in \mathcal{F}$ and its g_i-conjugate representation F^{g_i}.

Case 1. $\pi_i F = F$, i.e., $F \sim F^{g_i}$. Then by Clifford Theory, there are exactly $p := p_i$ pairwise nonequivalent irreducible extensions $D_0, \dots, D_{p-1}$ of F to G_i over K satisfying $D_k = \chi^k \otimes D_0$, where $1 = \chi^0, \chi^1, \dots, \chi^{p-1}$ are the characters of the cyclic group G_i/G_{i-1}. If D is such an extension, and $X := D(g_i)$, we necessarily have

(i) $X^{-1}F(h)X = D(g_i^{-1}hg_i) = F(g_i^{-1}hg_i) = F^{g_i}(h)$ for all $h \in G_{i-1}$, i.e., $X \in \operatorname{Int}(F^{g_i}, F)$ and

(ii) $X^p = F(g_i^p)$.

Conversely, it is easy to see that these two conditions are also sufficient in order to define an extension of F to G_i by $D(g_i) := X$.

By (i), $X = c \cdot X_{iF}$ for some $c \in K^*$. Now we use (ii) to determine c and obtain the equation

$$c^p X_{iF}^p = F(g_i^p).$$

This equation has p distinct solutions $c_0, \dots, c_{p-1}$ in K^*, which are (pe)th roots of unity. Thus $D_0, \dots, D_{p-1}$ are given by

$$D_k(g_i) := c_k X_{iF}.$$

As X_{iF} is monomial, each D_k is also monomial and obviously $\mathcal{T}_i$-adapted. Looking at the character of D_k, it is easy to see that the constants c_k are even eth roots of unity, so D_k is e-monomial. Hence, this part of the construction (in particular the computation of c_k) only involves calculations in the group of eth roots of unity.

Case 2. $\pi_i F \neq F$, i.e., $F \not\sim F^{g_i}$. Again by Clifford Theory, the induced representation $F \uparrow G_i$ is irreducible and $(F \uparrow G_i) \downarrow G_{i-1} = \bigoplus_{k=0}^{p-1} F^{g_i^k}$. As F is e-monomial, so is $F \uparrow G_i$. Now we have to adapt this induced representation to $\mathcal{T}_i$. To this end, note that $F^{g_i^k} \sim \pi_i^k F$ and $X_i := X_{i\pi_i^{k-1}F} \cdots X_{iF} \in \operatorname{Int}(F^{g_i^k}, \pi_i^k F)$. Now set $X := \bigoplus_k X_k$ and define the representation D by $D(a) := X(F \uparrow G_i)(a)X^{-1}$ for all $a \in G_i$. Obviously, D is irreducible and e-monomial. As each $\pi_i^k F \in \mathcal{F}$ is $\mathcal{T}_{i-1}$-adapted and

$$(D \downarrow G_{i-1})(b) = X(F \uparrow G_i \downarrow G_{i-1})(b)X^{-1} = \bigoplus_{k=0}^{p-1}(\pi_i^k F)(b)$$

for all $b \in G_{i-1}$, D is $\mathcal{T}_i$-adapted.

By these two constructions, we obtain all irreducible representations of G_i up to isomorphism, and Phase 1 is complete.

During the construction in Phase 1, we also build a bipartite graph in which $F \in \mathcal{F}$ and $D \in \mathcal{D}$ are linked if and only if F is a constituent of $D \downarrow G_{i-1}$. This "traceback" information will be be needed in the next phase.

Phase 2: Computation of τ_j and Y_{jD}. Let $F \in \mathcal{F}$ and $i < j \leq n$. We have to consider the same two cases as in Phase 1.

Case 1. $\pi_i F = F$. In Phase 1, we have computed the p extensions $D = D_0, \dots, D_{p-1}$ of F, where $D_k = \chi^k \otimes D$. As D is an extension of F, we have that $\tau_j D$ must be an extension of $\pi_j F$. Let $\Delta = \Delta_0, \dots, \Delta_{p-1}$ be the extensions of $\pi_j F$ computed in Phase 1 with $\Delta_k = \chi^k \otimes \Delta$. By Schur's lemma, $X_{jF} \in \mathrm{Int}(D_k^{g_j}, \tau_j D_k)$ for all k; hence we can set $Y_{jD_k} := X_{jF}$, although we do not know $\tau_j D_k$ yet. To determine the $\tau_j D_k$, observe that, for $k = 0$, the matrix Y_{jD} satisfies

$$Y_{jD} D^{g_j}(g_i) Y_{jD}^{-1} = (\tau_j D)(g_i) = \Delta_l(g_i) = \chi^l(g_i G_{i-1}) \cdot \Delta(g_i)$$

for a unique $0 \leq l < p$. To compute l, we only have to look at one nonzero entry on the left side of this e-monomial matrix equation and the corresponding entry on the right. With this l, set $\tau_j D := \Delta_l$. Once l is known, the remaining values $\tau_j D_k$ can be computed by "cyclic shifts" as follows: Let $0 < a_j < p$ be the unique integer such that $\chi^{g_j} = \chi^{a_j}$. (Note that a_j can be directly read off the pc-presentation, see §4.) Then for $0 < k < p$, we have

$$D_k^{g_j} = (\chi^k \otimes D)^{g_j} = (\chi^{g_j})^k \otimes D^{g_j} \sim \chi^{k a_j} \otimes \Delta_l.$$

Hence,

$$\tau_j D_k = \Delta_{l + k a_j \bmod p}.$$

Case 2. $\pi_i F \neq F$. In Phase 1, we have computed a $D \in \mathcal{D}$ such that $D \downarrow G_{i-1} = \bigoplus_{0 \leq k < p} F_k$ with $F_k := \pi_i^k F$ of degree f. Then $\tau_j D$ is the unique $\Delta \in \mathcal{D}$ such that $\Delta \downarrow G_{i-1}$ contains $\pi_j F$. This is already known from Phase 1. According to our construction, $\tau_j D \downarrow G_{i-1} = \bigoplus_{0 \leq k < p} \Phi_k$ with $\Phi_k := \pi_i^k \Phi$ for some $\Phi \in \mathcal{F}$.

From Phase 1, we know the unique permutation σ such that $\pi_j F_k = \Phi_{\sigma k}$ as well as e-monomial matrices $X_k := X_{jF_k} \in \mathrm{Int}(F_k^{g_j}, \Phi_{\sigma k})$.

We are looking for an e-monomial matrix $Y_{jD} \in \mathrm{Int}(D^{g_j}, \tau_j D)$. By Schur's lemma, Y_{jD} has the form[1]

$$Y_{jD} = \sigma \cdot \left(\bigoplus_{0 \leq k < p} c_k X_k \right)$$

for suitable constants $c_k \in K^*$. To determine the c_k, note that Y_{jD} must satisfy the equation

$$Y_{jD} D^{g_j}(g_i) Y_{jD}^{-1} = (\tau_j D)(g_i). \tag{3.1}$$

[1]For notational convenience, we are going to identify a permutation $\sigma \in \mathrm{Sym}(\{0, \dots, p-1\})$ with the pf-square block permutation matrix $P_\sigma \otimes E_f$, where P_σ is the p-square permutation matrix corresponding to σ and E_f denotes the f-square identity matrix.

As Y_{jD} is again uniquely determined up to a constant $c \in K^*$, we can set $c_0 := 1$.

According to our construction in Phase 1, there are e-monomial matrices A_k, $B_k \in K^{f\times f}$ such that $(\tau_j D)(g_i)$ is a block matrix of the form

$$(\tau_j D)(g_i) = (0, \dots, p-1) \cdot (B_0 \oplus \cdots \oplus B_{p-1}),$$

and $D^{g_j}(g_i)$ is a block matrix of the form

$$D^{g_j}(g_i) = \pi \cdot (A_0 \oplus \cdots \oplus A_{p-1})$$

for some $\pi \in \mathrm{Sym}(\{0, \dots, p-1\})$. Hence, (3.1) is equivalent to

$$\pi \cdot \left(\bigoplus_{k=0}^{p-1} c_{\pi k} X_{\pi k}\right) \left(\bigoplus_{k=0}^{p-1} A_k\right) \left(\bigoplus_{k=0}^{p-1} c_k^{-1} X_k^{-1}\right) = \sigma^{-1}(0, \dots, p-1)\sigma \cdot \left(\bigoplus_{k=0}^{p-1} B_{\sigma k}\right).$$

Because $c_0 = 1$, we can now successively determine $c_{\pi 0}, c_{\pi^2 0}, \dots$. Note that we obtain all c_k in this way since $\pi = \sigma^{-1}(0, \dots, p-1)\sigma$ is a p-cycle. To do this, it is sufficient to look at just one nonzero entry of $X_{\pi k} A_k c_k^{-1} X_k^{-1}$ and the corresponding entry in $B_{\sigma k}$.

4. Analysis

In order to analyze our algorithm, we have to describe its input and output more precisely.

The group G is given as a *power-commutator presentation* (also called an AG-system, see [5]). This is a presentation of the form[2]

$$\langle g_n, g_{n-1}, \dots, g_1 | g_i^{p_i} = u_i,\ 1 \le i \le n;\ g_i^{-1} g_j^{-1} g_i g_j = v_{ij},\ 1 \le i < j \le n\rangle$$

with primes p_i, and words $u_i = g_{i-1}^{a_{i,i-1}}, \dots, g_1^{a_{i,1}}$ and $v_{ij} = g_i^{b_{ij,i}}, \dots, g_1^{b_{ij,1}}$. Moreover, we require that the presentation be *consistent*, i.e., that every word in the generators has a unique normal form $g_n^{e_n} \cdots g_1^{e_1}$ with $0 \le e_k < p_k$. A consistent pc-presentation of this kind describes a supersolvable group. Moreover, $(G_n > G_{n-1} > \cdots > G_1 > \{1\})$, where $G_i := \langle g_i, \dots, g_1\rangle$ is a chief series of $G = G_n$. Conversely, every supersolvable group can be described by such a presentation.

The presentation of supersolvable groups by pc-presentations is of special interest for our purposes. A pc-presentation already contains all the information on the group needed in our algorithm, so no group operations are required at all.

With respect to this presentation, a monomial irreducible representation of the group G_i will be given by the representing matrices of the generators $g_i, \dots, g_1$.

As we have seen in the previous section, all nonzero entries of the monomial matrices that occur in our algorithm are eth roots of unity, where e is the exponent of G. Hence, we do not need general field arithmetic. Only symbolic computations in the group of eth roots of unity are required. Representing this group as $\mathbb{Z}/e\mathbb{Z}$, this just means integer arithmetic modulo e. For simplicity of our analysis, we assume that e is known. However, this is not necessary in

[2]Originally, pc-presentations have been defined to present solvable groups. We use a slightly modified form suited to the class of supersolvable groups.

practice: Just start with $e = 1$ and increase e by a factor of p whenever a pth root cannot be computed in Case 1 of our construction.

Obviously, most of the work in our algorithm consists of multiplying, inverting, and copying e-monomial matrices. Choosing a suitable data structure for e-monomial matrices is therefore vital for our algorithm's efficiency. We represent an n-square monomial matrix M in the form $M = \pi\,\mathrm{diag}(a_1, \dots, a_n)$ by its permutation structure $\pi \in S_n$ and its nonzero entries $a_1, \dots, a_n$. The permutation π is stored as a list $\pi(1), \dots, \pi(n)$ of integers. Under the realistic assumption that n and e fit into one standard one-word (e.g., 32-bit) integer on most machines, this representation uses $2n$ words of storage.

In order to make our analysis feasible, we are only going to count the following operations: arithmetic operations in $\mathbb{Z}/e\mathbb{Z}$, multiplication and inversion of permutations, and copying of e-monomial matrices. Other type of operations implied by our algorithm such as evaluating permutations, table lookups, and index calculations will not be counted, but it is clear that they will not affect the order of the algorithm's running time in a reasonable implementation.

To avoid separate counting of the different kinds of operations, we express them in terms of "basic operations" as follows, roughly reflecting the relation of the actual times taken by the different kinds of opertions on a typical computer:

- each arithmetic operation in $\mathbb{Z}/e\mathbb{Z}$ is counted as 1 basic operation;
- multiplication of two permutations in S_n or inversion of one permutation is counted as n basic operations;
- copying an n-square e-monomial matrix is counted as n basic operations.

Of course, these definitions are somewhat arbitrary, but they certainly match the actual running times within a constant factor. Hence again, we can be confident that our analysis determines the order of our algorithm's running time correctly.

In this model, the product of two n-square e-monomial matrices can be computed using $2n$ basic operations according to the following formula:

$$\pi\,\mathrm{diag}(a_1, \dots, a_n) \cdot \tau\,\mathrm{diag}(b_1, \dots, b_n) = \pi\tau\,\mathrm{diag}(a_{\tau(1)}b_1, \dots, a_{\tau(n)}b_n).$$

In a similar way, an n-square e-monomial matrix can be inverted in $2n$ basic operations.

Let us now analyze level i of our algorithm. In the sequel, $p := p_i$ and f denotes the degree of the representation F.

Phase 1: Computation of $\mathscr{D}$.

Case 1. $\pi_i F = F$.

X_{iF}^p can be computed with at most $2\log p \cdot 2f = 4f \log p$ operations using square-and-multiply.

$F(g_i^{p_i})$: From the pc-presentation, we know that $g_i^{p_i}$ is a word of the form $g_{i-1}^{e_{i-1}} \cdots g_1^{e_1}$. Using square-and-multiply to compute matrix powers, we can hence compute $F(g_i^{p_i}) = F(g_{i-1})^{e_{i-1}} \cdots F(g_1)^{e_1}$ with at most

$$2f\left(i - 2 + 2\sum_{k<i} \log p_k\right) = 4f \log|G_{i-1}| + 2f(i-2)$$

operations.

To obtain a solution c_0 of $c^p X_{iF}^p = F(g_i^p)$, it suffices to divide a single nonzero entry of $F(g_i^p)$ by the corresponding entry of X_{iF}^p and compute a pth root of the quotient. As computing a pth root just means an integer division in our symbolic representation, this can be done with only 2 operations.

In order to obtain the p extensions $D_0, \dots, D_{p-1}$ of F to G_i, we only have to compute the p matrices $D_k(g_i) = \omega^k c_0 X_{iF}$, where ω is a primitive pth root of unity. This takes $2pf$ operations.

Since $D_k \downarrow G_{i-1} = F$, the remaining representing matrices can be copied from level $i-1$, using $pf(i-1)$ operations.

Altogether, Case 1 takes at most

$$4f \log |G_i| + pf(i+1) + f(2i-4) + 2$$

operations.

Case 2. $\pi_i F \neq F$.

As $D \downarrow G_{i-1} = \bigoplus_{0 \le k < p} \pi_i^k F$, the matrices $D(g_{i-1}), \dots, D(g_1)$ are direct sums of matrices already computed in level $i-1$ and can be obtained by copying using $pf(i-1)$ operations.

The matrix $X = \bigoplus_k X_k$, where $X_k = X_{i\pi_i^{k-1}F} \cdots X_{iF}$, can be computed in at most $2(p-2)f + f = 2pf - 3f$ operations.

It remains to compute $D(g_i) = X(F \uparrow G_i)(g_i)X^{-1}$. This matrix has the form

$$\begin{aligned} D(g_i) &= \left(\bigoplus_{k=0}^{p-1} X_k\right)(0, \dots, p-1) \cdot (E_f \oplus \cdots \oplus E_f \oplus F(g_i^p))\left(\bigoplus_{k=0}^{p-1} X_k^{-1}\right) \\ &= (0, \dots, p-1) \cdot (X_1 X_0^{-1} \oplus \cdots \oplus X_{p-1}X_{p-2}^{-1} \oplus X_0 F(g_i^p) X_{p-1}^{-1}). \end{aligned}$$

As in Case 1, computing $F(g_i^p)$ takes at most $4f \log |G_{i-1}| + 2f(i-2)$ operations.

The inverses of $X_1, \dots, X_{p-1}$ can be computed with at most $2(p-1)f$ operations. (Note that X_0 is the identity matrix.)

Finally, we have to compute p products of f-square e-monomial matrices to obtain $D(g_i)$ according to the above formula. This takes at most $2pf$ operations.

Altogether, Case 2 takes at most $4f \log |G_{i-1}| + pf(i+5) + f(2i-9)$ operations for the p f-dimensional representations $\pi_i^k F$ of G_{i-1}, that is,

$$\frac{4}{p} f \log |G_{i-1}| + f(i+5) + \frac{f}{p}(2i-9)$$

operations each.

Phase 2: Computation of τ_j and Y_{jD}. Let $F \in \mathcal{F}$ and $i < j \le n$.

Case 1. $\pi_i F = F$. For each $i < j \le n$, we have to do the following:

The matrices $Y_{jD_k} = X_{jF}$ $(0 \le k < p)$ are copies from level $i-1$ in pf operations.

To compute l, we have to compute one nonzero entry of $Y_{jD} D^{g_j}(g_i) Y_{jD}^{-1}$ and divide it by the corresponding entry of $\Delta(g_i)$.

First, we compute $D^{g_j}(g_i) = D(g_j^{-1} g_i g_j)$: From the pc-presentation, we know that $g_j^{-1} g_i g_j$ is a word of the form $g_i^{e_i} \cdots g_1^{e_1}$. Hence, $D^{g_j}(g_i)$ can be computed in at most $4f \log |G_i| + 2f(i-1)$ operations (see Case 1 of Phase 1).

Next, we compute one nonzero entry of $Y_{jD}D^{g_j}(g_i)Y_{jD}^{-1}$ with 3 more operations.

To obtain l, we divide this entry by the corresponding entry of $\Delta(g_i)$ and compute the pth root of the quotient, which is a pth root of unity. As this just means an integer division in our symbolic representation, it takes 2 more operations to compute l.

Now the $\tau_j D_k$ can be obtained by "cyclic shifts", which are free of cost in our model.

Altogether, Case 1 takes at most

$$4f\log|G_i| + pf + f(2i-2) + 5$$

operations.

Case 2. $\pi_i F \neq F$. For each $i < j \leq n$, we have to do the following:

$\tau_j D$ can be determined free of cost by table lookups.

Next we determine the coefficients c_k, $0 < k < p$. For each coefficient, we have to compute one nonzero entry of $X_{\pi k}A_k c_k^{-1}X_k^{-1}$ and divide it by the corresponding entry of $B_{\sigma k}$. Altogether, this takes $5p-5$ operations.

Finally, in order to obtain Y_{jD}, we have to compute $c_2X_2, \ldots, c_{p-1}X_{p-1}$. This takes at most $2(p-1)f$ operations.

Altogether, Case 2 takes at most $2pf - 2f + 5p - 5$ operations for p f-dimensional representations of G_{i-1}, that is,

$$2f - 2\frac{f}{p} + 5 - f\frac{5}{p}$$

operations each.

Altogether, we have proved the following.

Lemma 4.1. *The number of operations taken to process one f-dimensional representation of G_{i-1} in both phases is at most*

$$8f\log|G_i| + pf(i+2) + f(4i-6) + 7$$

for Case 1, *and at most*

$$\frac{4}{p}f\log|G_{i-1}| + f(i+7) + \frac{f}{p}(2i-11) + 5 - \frac{5}{p}$$

for Case 2.

As the first case is obviously more expensive than the second, our worst-case analysis will be based on Case 1. If we sum up over all representations $F \in \mathcal{F}$ and use the fact that $\sum_{F\in\mathcal{F}} \deg F \leq |G_{i-1}|$, we obtain the upper bound

$$\begin{aligned} &8|G_{i-1}|\log|G_i| + |G_i|(i+2) + |G_{i-1}|(4i+1) \\ &\leq 4|G_i|\log|G_i| + |G_i|(i+2) + |G_i|(2i+1/2) \\ &\leq 4|G_i|\log|G_i| + 3|G_i|i + 2.5|G_i| \\ &\leq 7|G_i|\log|G_i| + 2.5|G_i| \end{aligned}$$

for the number of operations in level i of the algorithm.

Summing up over all levels $1 \leq i \leq n$, we obtain the upper bound

$$\begin{aligned} 7\sum_{i=1}^{n}|G_i|\log|G_i| + 2.5\sum_{i=1}^{n}|G_i| \\ &\leq 7\log|G_n|\sum_{i=1}^{n}|G_i| + 2.5\sum_{i=1}^{n}|G_i| \\ &\leq 7\log|G_n|\sum_{i=1}^{n}|G_n|2^{i-n} + 2.5\sum_{i=1}^{n}|G_n|2^{i-n} \\ &\leq 7|G_n|\log|G_n|\sum_{i=0}^{n-1}2^{-i} + 2.5|G_n|\sum_{i=0}^{n-1}2^{-i} \\ &< 14|G_n|\log|G_n| + 5|G_n| \end{aligned}$$

for the total number of operations of our algorithm, and we have proved the following.

Theorem 4.1. *The ordinary irreducible representations of a supersolvable group G can be computed from a power-commutator presentation of G with at most*

$$14|G|\log|G| + 5|G| = O(|G|\log|G|)$$

basic operations.

Observe that the output of our algorithm consists of $2n\sum_{D\in\mathscr{D}}\deg D$ integer numbers, where $\mathscr{D}$ denotes the set of irreducible representations of G computed in level n. Obviously, this is at most $2|G|\log|G|$. On the other hand, this upper bound is sharp for abelian 2-groups. Although it does not prove a lower complexity bound, this indicates that our algorithm is (in terms of the group order) *optimal* up to a constant factor.

We have not yet implemented our algorithm. However, it can be expected to run substantially faster than the old algorithm, for which some running times are given in [2].

Bibliography

1. L. Babai and L. Rónyai, *Computing irreducible representations of finite groups*, Proc. 30th IEEE Sympos. Foundations of Comput. Science, IEEE Computer Society Press, Los Alamitos, CA, 1989, pp. 93–98.
2. U. Baum, *Existence and efficient construction of fast Fourier transforms on supersolvable groups*, Comput. Complexity **1/3** (1992), 235–256.
3. M. Clausen, *Fast generalized Fourier transforms*, Theoret. Comput. Sci. **67** (1989), 55–63.
4. A. Fässler and E. Stiefel, *Group theoretical methods and their applications*, Birkhäuser, Boston, 1992.
5. C. R. Leedham-Green, *A soluble group algorithm*, Computational Group Theory (M. D. Atkinson, ed.), Academic Press, London, 1984, pp. 85–101.

Institut für Informatik, Universität Bonn, Römerstr. 164, 53117 Bonn, Germany
E-mail address: uli@leon.cs.uni-bonn.de
E-mail address: clausen@leon.cs.uni-bonn.de