J. Symbolic Computation (2002) **11**, 1–000

Symmetry-Based Matrix Factorization

SEBASTIAN EGNER[†] AND MARKUS PÜSCHEL[‡§]

[†] *Philips Research, 5656 AA Eindhoven, The Netherlands*
[‡] *Department of Electrical and Computer Engineering, Carnegie Mellon University*

(Received 15 January 2002)

We present a method to factor a given matrix M into a short product of sparse matrices, provided M has a suitable "symmetry". This sparse factorization represents a fast algorithm for the matrix-vector multiplication with M. The factorization method consists of two essential steps. First, a combinatorial search is used to compute a suitable symmetry of M in the form of a pair of group representations. Second, the group representations are decomposed stepwise, which yields factorized decomposition matrices and determines a sparse factorization of M. The focus of this article is the first step, finding the symmetries. All algorithms described have been implemented in the library AREP. We present examples for automatically generated sparse factorizations—and hence fast algorithms—for a class of matrices corresponding to digital signal processing transforms including the discrete Fourier, cosine, Hartley, and Haar transform.

1. Introduction

In this article we address the following fundamental problem: "Given a not necessarily square matrix M, construct an algorithm to evaluate the linear mapping $x \mapsto M \cdot x$ with as few arithmetic operations in the base field as possible". We present an algorithm that takes a given matrix M as input, and outputs a factorization of M into a short product of highly structured sparse matrices,

$$M = M_1 \cdot M_2 \cdots M_k, \quad M_i \text{ sparse}.$$

By "short", we mean that this factorization actually reduces the cost (in terms of the number of additions and multiplications) of computing $M \cdot x$. In the paper we will use interchangeably "sparse factorization of M"and "fast algorithm for M".

Our method is applicable if and only if the matrix M has a "symmetry" in a sense being defined. Intuitively, the symmetry captures redundancy in M given by linear relationships among the entries of M. Then, we use the symmetry to derive a sparse factorization. The factorization method consists of three steps: (1) Find a suitable symmetry of M by combinatorial search. The symmetry is given as a pair of group representations of a common finite group. (2) Decompose both representations recursively into a direct sum of irreducible representations. This yields decomposition matrices that are products of sparse

§ The work of Markus Püschel was supported in part by NSF through award 9988296.

0747–7171/90/000000 + 00 \$03.00/0

matrices. (3) Find a correction matrix that reconstructs M from these decomposition matrices. The correction matrix is sparse.

The first step—and the method in general—is the focus of this article. In particular, it is explained what we mean by symmetry, what types of symmetry have proven useful for our purposes and how these symmetries can be found algorithmically. The second step is concerned with the constructive decomposition of representations and is explained in detail in Püschel (2002). For the sake of completeness we give a brief survey of these methods. The third step of the method involves only matrix multiplications.

We will give several examples of matrices where the method can be applied and indeed constructs an efficient algorithm. The examples are chosen from the field of digital signal processing and include the discrete Fourier transform, cosine transforms, and the Hartley transform. It was our original motivation to construct fast algorithms for these kind of matrices but the method itself is not restricted to discrete signal transforms.

1.1. Background

The factorization method has its roots in the relationship between the discrete Fourier transform (DFT) used in digital signal processing, and the theory of group representations. In signal processing, the DFT is defined as a multiplication of a (complex) vector $x \in \mathbb{C}^n$ (the sampled signal) by the DFT matrix of size $n \times n$, given by

$$\mathrm{DFT}_n = [e^{2\pi i k\ell/n} \mid 0 \leq k, \ell \leq n-1], \quad i = \sqrt{-1}.$$

In the framework of representation theory, the DFT_n can be viewed as the isomorphism decomposing the group algebra $\mathbb{C}[\mathsf{Z}_n]$ of a cyclic group Z_n into a direct sum of algebras of dimension 1, if suitable, canonical bases are chosen,

$$\mathrm{DFT}_n : \ \mathbb{C}[\mathsf{Z}_n] \to \mathbb{C} \oplus \ldots \oplus \mathbb{C}.$$

This decomposition is a special case of a theorem first proved by Wedderburn (1907) in his classification of semisimple algebras. Let G be a finite group and $d_1, \ldots, d_h$ the degrees of a complete set of irreducible representations. Then the group algebra $\mathbb{C}[G]$ is isomorphic to the direct sum of simple algebras $\mathbb{C}^{d_1 \times d_1} \oplus \cdots \oplus \mathbb{C}^{d_h \times d_h}$. Based on this, Apple and Wintz (1970) generalize *the DFT* for $G = \mathsf{Z}_n$ to a DFT for a general abelian group G and Karpovsky and Trachtenberg (1977) generalize to arbitrary finite groups.

The algebraic description has proven extremely useful in deriving and understanding the structure of fast algorithms for the DFT of a group G. In essence, the structure of a fast algorithm reflects the structure of an associated representation of $\mathbb{C}[G]$. Most important for applications, Auslander *et al.* (1984) and Beth (1984) derive and explain the famous Cooley-Tukey algorithm (Cooley and Tukey, 1965)—in signal processing known as the "fast Fourier transform", or FFT—by a stepwise decomposition of $\mathbb{C}[\mathsf{Z}_n]$.

For an introduction and survey of the area of DFTs for groups G and their fast algorithms, we refer the reader to the textbook by Clausen and Baum (1993) or the more recent survey article by Maslen and Rockmore (1995). Maslen and Rockmore (1997) give a recent overview on the complexity of evaluating the DFT for several classes of groups including the more general view of the DFT on "homogeneous spaces" $\mathbb{C}[G/K]$, $K \leq G$.

Despite the success of the DFT, it became more and more important to find efficient algorithms for other signal transforms as well. One important example is the discrete cosine transform (DCT) used in the JPEG compression standard for digital images.† Unfortunately, the DCT (and its many variants), and most other transforms used in signal processing, cannot be interpreted as generalized DFTs. This naturally posed the question whether it is possible to characterize these transforms in the framework of group representations, and, in the affirmative case, use this connection to derive their fast algorithms.

To answer this question, Minkwitz (1993) reversed the way of working. Instead of defining a transform based on some given algebraic structure (a finite group or a homogeneous space), he tried to find the algebraic structure of a *given transform*. For this purpose, he defined the notion of "symmetry", which associates to a given matrix a pair of matrix representations of a common group. These symmetries have to be explicitly found since they are not known by construction (as for a generalized DFT). Methods to find symmetry in a matrix are the focus of this article, which is based on the work of Egner (1997). Once a symmetry is explicitly known, as representations of finite groups, it must be decomposed into irreducible components such that the decomposition is stepwise and constructive. Minkwitz (1993) gave algorithms to accomplish this for permutation representations of solvable groups by using Clifford's theory. Püschel (2002) generalized the methods to monomial representations of solvable groups. Together, the algorithms for finding symmetry and for decomposing monomial representations constructively form a powerful algorithm to factor a given matrix into a short product of sparse matrices. Application to signal transforms yields fast algorithms in many cases, which shows that the connection between signal processing and representation theory is stronger than previously understood.

1.2. Structure of this Article

In Section 2 we introduce the notion of symmetry of a matrix M, and explain which types of symmetry are useful to derive a sparse factorization of M. Then we explain the algorithm for the symmetry-based matrix factorization. The problem of finding a suitable symmetry is treated in Section 3. The subsections are devoted to the different types of symmetry considered. The second major ingredient for deriving a sparse matrix factorization is a method to decompose monomial representations, which is explained in Section 4. The software library AREP contains all presented algorithms and is briefly explained in Section 5. Finally, Section 6 contains a gallery of automatically generated sparse factorizations for a class of signal transform matrices, including runtime measurements of the factorization algorithm.

1.3. Notation

For the convenience of the reader, we give a brief overview on notation and concepts from ordinary representation theory of finite groups. For further information refer to standard textbooks such as Curtis and Reiner (1962).

Matrices A are introduced by specifying the entry $A_{k,\ell}$ at position k, ℓ over some index range, as in $A = [A_{k,\ell} \mid 0 \leq k, \ell \leq n-1]$. The operators $\oplus, \otimes$ are used for the direct sum and tensor (or Kronecker) product of matrices, respectively. The $(n \times n)$-permutation matrix corresponding to the permutation σ is denoted by $[\sigma, n] = [\delta_{i^\sigma j} \mid 1 \leq i, j \leq n]$

† JPEG is part of each and every Internet browser.

or simply by σ if the matrix size n is known from the context. A monomial matrix has exactly one non-zero entry in each row and column (and is hence invertible) and is written as $[\sigma, L] = [\sigma, \text{length}(L)] \cdot \text{diag}(L)$, where $\text{diag}(L)$ is a diagonal matrix with the list L on the diagonal. The $(n \times n)$-identity matrix is denoted by $\mathbf{1}_n$. Finally, we use $\omega_n = e^{2\pi i/n}$ for the complex primitive nth root of unity.

A (matrix) *representation* of a finite group G of degree $\deg(\phi) = n$ is a homomorphism $\phi : G \to \text{GL}_n(\mathbb{F})$ from G into the group of invertible $(n \times n)$-matrices over a field, which we denote by the letter $\mathbb{F}$ throughout this article. A representation ϕ is a *permutation* or *monomial* representation, if all images $\phi(g)$ are permutation matrices or monomial matrices, respectively. We denote by $1_G : g \mapsto 1$ the *trivial representation* of G (of degree 1). If $A \in \text{GL}_n(\mathbb{F})$, then $\phi^A : g \mapsto A^{-1} \cdot \phi(g) \cdot A$ is the *conjugate* of ϕ by A. ϕ and ψ are called *equivalent*, if $\phi = \psi^A$. If ϕ, ψ are representations of G, then the representation $\phi \oplus \psi : g \mapsto \phi(g) \oplus \psi(g) = \left[\begin{smallmatrix} \phi(g) & 0 \\ 0 & \psi(g) \end{smallmatrix}\right]$ is called the *direct sum* of ϕ and ψ. ϕ is called *irreducible*, if it cannot be conjugated into a direct sum. In this paper, we will deal only with *ordinary* representations. This means the characteristic of $\mathbb{F}$ does not divide the group order $|G|$ (Maschke condition). In this case, every representation ϕ can be conjugated, by a suitable matrix A, into a direct sum of irreducible representations ρ_i (Maschke's theorem). In other words, $\phi^A = \bigoplus_{i=1}^k \rho_i$, which is called a *decomposition* of ϕ and A is referred to as a *decomposition matrix* for ϕ.

Let $H \leq G$ be a subgroup and $T = (t_1, \ldots, t_k)$ a transversal, meaning a system of representatives of the right cosets of H in G. Furthermore, let ϕ be a representation of H. We define the *induction* of ϕ to G with respect to T as the representation of G that is defined as

$$(\phi \uparrow_T G)(g) = [\dot{\phi}(t_i g t_j^{-1}) \mid i, j],$$

where $\dot{\phi}(x) = \phi(x)$ for $x \in H$ and $\dot{\phi}(x)$ is the all-zero matrix if $x \notin H$. A regular representation is an induction of the form $\phi = (1_E \uparrow_T G)$ where E denotes the trivial subgroup of G. If ϕ is a representation of G then $\phi \downarrow H$ denotes the *restriction* of ϕ to H. The *intertwining space* of the representations ϕ, ψ is defined as the vector space of matrices $\mathsf{Int}(\phi, \psi) = \{M \in \mathbb{F}^{\deg(\phi) \times \deg(\psi)} \mid \phi(g) \cdot M = M \cdot \psi(g),\ g \in G\}$.

Finally, we would like to emphasize that we are working with *matrix* representations, and not with equivalence classes of representations, for which characters are the appropriate data structure.

2. Symmetry-Based Matrix Factorization

In this section we introduce the notion of "symmetry" of a matrix M and explain how it is used to factor M into a product of sparse matrices. The factorization represents a fast algorithm for computing the matrix-vector product $M \cdot x$.

The symmetry of a matrix serves a two-fold purpose. First, it captures redundancy contained in the matrix that arises from relationships among the entries of M. Second, it establishes a connection between the matrix and certain group representations. This connection then is used to factorize M.

The origin of the following definition is due to Minkwitz (1993).

Definition 2.1 Let $M \in \mathbb{F}^{n \times m}$ be a rectangular matrix over a base field $\mathbb{F}$. We call a

Table 1. Mnemonic names for types of a representation ϕ.

perm	permutation representation
mon	monomial representation
block	permuted direct sum: $(\phi_1 \oplus \cdots \oplus \phi_r)^\pi$, where π is a permutation
irred	like *block* but all the ϕ_i are irreducible
mat	any matrix representation under Maschke condition

pair (ϕ, ψ) of representations of the same group G a *symmetry* of M if

$$\phi(g) \cdot M = M \cdot \psi(g) \quad \text{for all } g \in G.$$

We write this symbolically as $\phi \xrightarrow{M} \psi$.

The definition implies the rules

$$\phi \xrightarrow{M_1} \psi \xrightarrow{M_2} \rho \quad \Rightarrow \quad \phi \xrightarrow{M_1 \cdot M_2} \rho, \quad \text{and}$$
$$\phi \xrightarrow{M} \psi \quad \Rightarrow \quad \psi \xrightarrow{M^{-1}} \phi, \quad \text{for } M \text{ invertible.}$$

Note that M has the symmetry (ϕ, ψ) if and only if $M \in \mathsf{Int}(\phi, \psi)$, the intertwining space of ϕ and ψ (defined above). Equivalently, we can formulate a symmetry (ϕ, ψ) of M as the invariance of M under the operation $\bullet$ of G on $\mathbb{F}^{n \times m}$, given by

$$g \bullet M = \phi(g) \cdot M \cdot \psi(g^{-1}), \quad g \in G,\ M \in \mathbb{F}^{n \times m}.$$

The purpose of the group G is to link the two representations ϕ and ψ together, but G has also a major influence on the structure of the factorization for M obtained (see Step 2 of the factorization algorithm in this section and Section 4).

Definition 2.1 is in its generality hardly useful to capture redundancy contained in the matrix M. For example, if M is invertible, then (ϕ, ϕ^M) is a symmetry of M for every group representation ϕ of suitable degree. And indeed, not all matrices can have a useful sparse factorization! Consider the following crude estimate of the algebraic problem complexity: If we derive an $O(n \log n)$-algorithm for an $(n \times n)$-matrix then there are only $O(n \log n)$ degrees of freedom in the algorithm but $\mathbb{F}^{n \times n}$ is a n^2-dimensional vector space.

However, if ϕ and ψ are restricted to certain types of representations (for example permutation or direct sum of irreducibles) then the intertwining space $\mathsf{Int}(\phi, \psi)$ does become interesting. This has led Minkwitz (1993) to study different *types* of symmetry arising from different *types* of representations ϕ, ψ. For convenience we use mnemonic names to describe these different types. For example, ϕ is of type *mon* if it is monomial, or of type *irred*, if it is a direct sum of irreducible representations (possibly conjugated by a permutation). A full list of the considered types is listed in Table 1. Correspondingly we name the types of symmetry. For example, (ϕ, ψ) is a *perm-irred* symmetry if ϕ is of type *perm* and ψ is of type *irred*. The reason for considering these types will become clear in the following.

We will now describe the method to construct a fast algorithm—represented as a sparse factorization—for the matrix-vector multiplication with a given matrix M. The method is displayed in Figure 1 and consists of the following three steps.

Step 1: Finding symmetry

Step 2: Decomposing representations

Step 3: Combining decompositions

$$\begin{tikzcd} \phi \arrow[r, "M"] \arrow[d, "A = A_1 \cdots A_r"'] \arrow[dr, phantom, "///"] & \psi \arrow[d, "B = B_1 \cdots B_s"] \\ \widehat{\phi} \arrow[r, "C"] & \widehat{\psi} \end{tikzcd}$$

Figure 1. Factorizing the matrix M using a suitable symmetry (ϕ, ψ). The factorization is given by $M = A_1 \cdots A_r \cdot C \cdot B_s^{-1} \cdots B_1^{-1}$; the A_i, C, and the B_j^{-1} are all sparse.

Step 1: Finding Symmetry

The goal of this step is to make the symmetry in the matrix M explicit in the sense that the pair of representations is actually known by group generators and their images. Given the matrix M it is first decided which type of symmetry to use for constructing a factorization of M (e.g., perm-irred symmetry or mon-mon symmetry). Then a combinatorial search is run on M for the chosen type of symmetry as described in Section 3. The result is a pair (ϕ, ψ) of representations of the chosen types such that $\phi \xrightarrow{M} \psi$.

As we will see in Step 3, a symmetry (ϕ, ψ) is useful for factorizing M only if for both representations ϕ and ψ a decomposition matrix can be determined as a product of sparse matrices. With our current methods this can be done for representations of type *irred* (the decomposition matrix is a permutation) and of type *mon* (of solvable groups, see Section 4). Thus, the following types of symmetry are of interest: mon-mon, mon-irred, irred-mon, and the subtypes perm-perm, perm-irred, irred-perm. We omit the type irred-irred, since it is not of practical importance (in general, it requires M to be already sparse—a consequence of Schur's lemma). Since transposing a matrix with irred-mon (or irred-perm) symmetry yields a matrix with mon-irred (or perm-irred) symmetry, we restrict our investigations to the latter types.

Finding mon-mon and mon-irred symmetry requires substantially different approaches (see Section 3). Note that matrices may have both types of symmetry, which leads to different factorizations (e.g., Section 6.4).

Step 2: Decomposing Representations

The second step decomposes the representations ϕ and ψ into a direct sum of irreducibles $\widehat{\phi}$ and $\widehat{\psi}$ with decomposition matrices A and B, respectively, i.e.,

$$\phi^A = \widehat{\phi} = \phi_1 \oplus \cdots \oplus \phi_k, \quad \text{and} \quad \psi^B = \widehat{\psi} = \psi_1 \oplus \cdots \oplus \psi_\ell, \quad \phi_i, \psi_j \text{ irreducible.}$$

The crucial point is that A and B are determined as (short) product of sparse matrices, $A = A_1 \cdots A_r$, and $B = B_1 \cdots B_s$. For the type *irred* this product reduces to a single permutation; for the type *mon* (or *perm*) the product is obtained through a decomposition algorithm that recurses over the structure of the representation (see Section 4). As a simple example, if ϕ is recognized as a permuted direct sum then it is sufficient to decompose the direct summands independently. The structural recursion approach is neither the one used by Parker (1984) for the MeatAxe, nor does it just evaluate projections onto irreducibles. The principle of the MeatAxe is to choose "random" elements x of the group algebra and decompose the full vector space into x-invariant subspaces until the components are irreducible. Unfortunately, this method does not produce sparse matrices to decompose the representation. The recursive method to decompose monomial representations, as used here, is briefly explained in Section 4. A more detailed treatment is beyond the scope of this article and we refer the reader to Püschel (2002).

Step 3: Combining Decompositions

The final step is trivial but important. It computes the matrix

$$C = A^{-1} \cdot M \cdot B$$

to make the diagram in Figure 1 commute. The matrix C is in the intertwining space $\mathsf{Int}(\widehat{\phi}, \widehat{\psi})$, which implies—using Schur's lemma—that C contains zeros at all components that connect inequivalent representations ϕ_i and ψ_j. So, C is permuted block diagonal with the sizes and positions of the blocks depending on the irreducibles contained in ϕ and ψ. For example, if all irreducibles ϕ_i in ϕ are pairwise inequivalent and ψ is equivalent to ϕ then there are at most $\sum_i (\deg \phi_i)^2$ non-zero entries in C. Finally, we note that the type of sparse matrices A_i, B_j generated by the decomposition algorithm for monomial representations preserves its sparsity under inversion.

Taken together, we read from Figure 1 the following sparse factorization of M:

$$M = A \cdot C \cdot B^{-1} = A_1 \cdots A_r \cdot C \cdot B_s^{-1} \cdots B_1^{-1}.$$

From a different point of view, our factorization method can be viewed as a particular type of common subexpression elimination for matrix-vector multiplication algorithms. The common subexpressions are captured by the respective symmetry.

3. Finding Symmetries of a Matrix

We now turn to the problem of actually finding symmetry, given a matrix M. For this purpose it is useful to look at symmetry in a slightly different way: We consider individual pairs (L, R) of invertible matrices such that $LM = MR$. Clearly, this property is retained under componentwise multiplication and inversion: If $LM = MR$ and $L'M = MR'$ then $LL'M = MRR'$ and $L^{-1}M = MR^{-1}$. Hence, all pairs (L, R) of invertible matrices satisfying $LM = MR$ form a group under componentwise multiplication. We could call it the "universal symmetry group" of M because it contains all more specific symmetries as a subgroup.

Unfortunately, as mentioned earlier, the universal symmetry is not very helpful to decompose the matrix M. For example, if M is invertible, the universal symmetry is the set of all pairs $(L, M^{-1}LM)$ with invertible L; the universal symmetry group is (isomorphic to) the $\mathrm{GL}_n(\mathbb{F})$. Clearly, this symmetry does not provide information about M. There are two further problems with general symmetries: First, the group might not be finite and thus the symmetry group can not be computed, and, second, we do not know an algorithm to decompose a general representation into irreducibles such that the decomposition matrix is a product of sparse matrices.

Therefore, we introduce restrictions on the matrices L and R, which leads, as explained in Section 2, to the types of symmetry considered in this paper: mon-irred symmetry and mon-mon symmetry, and the subtypes perm-irred symmetry and perm-perm symmetry.

The task of finding the symmetry of a certain type for a given matrix M can be described as finding a generating set for the group G of all pairs (L, R) of invertible matrices of the given type such that $LM = MR$. We call such a group a *symmetry group*, although it is very important that it is not just an abstract group but a group of pairs of matrices. The relation of this view of symmetry to the description in Section 2 is as follows: Let (ϕ, ψ) be a pair of representations of a group $\tilde{G}$. Then $G = \{(\phi(g), \psi(g)) \mid g \in \tilde{G}\}$ is a group of pairs (L, R) of matrices such that $LM = MR$. Conversely, let G be a group of matrices (L, R) such that $LM = MR$. Then the canonical projections $\Pi_1 = (L, R) \mapsto L$ and $\Pi_2 = (L, R) \mapsto R$ are representations of the group G and the pair (Π_1, Π_2) is a symmetry of M in the sense of Section 2. Note that $\tilde{G}$ may be larger than G, i.e., G may be a homomorphic image of $\tilde{G}$, because ϕ and ψ could both map some normal subgroup of $\tilde{G}$ into the trivial group. For the purpose of decomposing M the knowledge of G is then as good as the knowledge of $\tilde{G}$.

In the remainder of this section we first explain, in Section 3.1, the general structure of an arbitrary symmetry group, and how it can be used to simplify the symmetry search. Sections 3.2–3.6 are then devoted to finding the different types of symmetry. Section 3.2 deals with the perm-perm symmetry, and Section 3.3 shows how the mon-mon symmetry (of a certain class) of a matrix can be found via the perm-perm symmetry of a suitable larger matrix. Section 3.4 is concerned with finding the perm-mat symmetry, which is used as a subroutine for finding the perm-irred symmetry in Section 3.5. Finding the mon-irred symmetry uses a similar approach, which is sketched in Section 3.6.

3.1. Subdirect Product Structure of the Symmetry Group

The structure of an arbitrary symmetry group is given by the *subdirect product* of its left and right projections $\Pi_1(G)$ and $\Pi_2(G)$ (with respect to a certain isomorphism). We detail this structure formally and explain how it is used to simplify the symmetry search.

First we recall the notion of a *subdirect product with identified factor groups* (Huppert, 1983, Kap. I, §9). Let G_1, G_2 be groups with normal subgroups $N_1 \trianglelefteq G_1$ and $N_2 \trianglelefteq G_2$, and assume $\varphi : G_1/N_1 \to G_2/N_2$ is an isomorphism of the factor groups. Then

$$G_1 \curlywedge G_2 = \{(g_1, g_2) \in G_1 \times G_2 \mid \varphi(g_1 N_1) = g_2 N_2\}$$

is a subgroup of the direct product $G_1 \times G_2$, called the subdirect product with identified factor groups†.

Lemma 3.1 *Let M be a matrix and G be a group of pairs (L, R) of matrices such that $LM = MR$. Furthermore, let Π_1 and Π_2 denote the canonical projection from G onto the first and the second component, respectively. Then G is the subdirect product $\Pi_1(G) \curlywedge \Pi_2(G)$ with identified isomorphic factor groups*

$$\Pi_1(G)/\Pi_1(\ker \Pi_2) \cong \Pi_2(G)/\Pi_2(\ker \Pi_1).$$

Proof. The lemma can be shown by checking the definition of the subdirect product for the isomorphism φ defined by

$$\varphi(L' \cdot \{L \mid (L, 1) \in G\}) = R' \cdot \{R \mid (1, R) \in G\},$$

for all $(L', R') \in G$. □

The subdirect structure can be used to simplify the search for symmetry. In the first step the normal subgroups of the left and right projection are constructed. Then, in the

† The symbol $\curlywedge$ depicts the two towers $G_i \trianglerighteq N_i \geq E$ with factor groups G_i/N_i, $i = 1, 2$, identified. Here E denotes the trivial subgroup.

second step, the common factor group is constructed. We illustrate this approach by an example for the perm-perm symmetry. Assume we want to find the group G of all pairs (L, R) of permutation matrices such that $LM = MR$ for the matrix

$$M = \begin{bmatrix} 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \end{bmatrix}.$$

By definition, $N_1 = \Pi_1(\ker \Pi_2) = \{L \mid LM = M\}$ is the group of all permutations L of the rows of M that leave M invariant. Since the third and fourth row are equal, $N_1 = \langle(3,4)\rangle$. In the same way $N_2 = \{R \mid M = MR\} = \langle(1,4)\rangle$, because the first and fourth column of M are equal. (We write permutation matrices as permutations in cycle notation assuming the matrix size is known from the context.) Now we reduce the matrix M by partitioning its rows as $(1|2|3\,4)$ and partitioning its columns as $(1\,4|2|3)$, i.e., the double rows and columns are removed. (We denote partitions by listing the elements, separating the blocks with a vertical bar.) This leaves us with the smaller matrix

$$\tilde{M} = \begin{bmatrix} 1 & 0 & 1 \\ 1 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix}.$$

The second step involves finding the common factor group of the projections of G. The factor group acts by permuting rows and columns of $\tilde{M}$, which in turn correspond to blocks of rows and columns of M. It is readily seen that the only non-trivial symmetry operation on $\tilde{M}$ is $(1,2) \cdot \tilde{M} = \tilde{M} \cdot (2,3)$. Hence, the factor groups (with rows, columns named as for M) are $G_1/N_1 = \langle(1,2)\rangle \cong G_2/N_2 = \langle(2,3)\rangle$. As the result we obtain the symmetry group $G = \langle((3,4),1),(1,(1,4)),((1,2),(2,3))\rangle$.

The subdirect structure can also be exploited for types of symmetry other than perm-perm. If the left representation is to be monomial (i.e, of type *mon*), then the group $N_1 = \{L \mid LM = M\}$ contains all monomial transformations on rows that are scalar multiples of each other. Similarly, if the left representation is unrestricted (i.e., of type *mat*) then N_1 contains a general linear group acting on the null space $\{x \mid xM = 0\}$. Similar statements hold for the right representation, depending on its type. In each case it is possible to reduce the search of the symmetry by reducing the matrix M to a matrix $\tilde{M}$ for which the corresponding groups $\tilde{N}_1$ and $\tilde{N}_2$ are trivial. This reduction is mathematically trivial, although the bookkeeping is rather involved and complicates implementation. We omit these details.

3.2. Perm-Perm Symmetry

The simplest type of symmetry we consider is the *perm-perm* symmetry. Given a matrix $M \in \mathbb{F}^{n \times m}$, define the group of pairs of permutations

$$\mathsf{PermPerm}(M) = \{(L, R) \in \mathsf{S}_n \times \mathsf{S}_m \mid LM = MR\},$$

where S_n denotes the symmetric group permuting n elements. By abuse of language we will frequently drop the distinction between permutation and permutation matrix. Using Lemma 3.1 it is sufficient to consider matrices M with pairwise distinct rows and pairwise distinct columns.

An example of the perm-perm symmetry is the well known symmetry of the discrete Fourier transform (we use the shorthand notation $i \mapsto f(i)$ to denote the permutation):

Lemma 3.2 *Let* $\mathrm{DFT}_n = [e^{2\pi j k\ell/n} \mid 0 \leq k, \ell < n]$ *and define* $L_k = (i \mapsto ki \bmod n)$ *for each* $k \in Z_n^\times = \{k \mid \gcd(k,n) = 1\}$. *Then*

$$\mathsf{PermPerm}(\mathrm{DFT}_n) = \{(L_k, L_k^{-1}) \mid k \in Z_n^\times\}.$$

Unfortunately, a polynomial time algorithm for constructing $\mathsf{PermPerm}(M)$ for a given matrix M implies a polynomial time algorithm for testing isomorphism of two given graphs (Egner, 1997, Satz 3.2). The graph isomorphism problem is well studied ([OPEN1] from Garey and Johnson (1979)) and no polynomial time algorithm is known for it (nor has it been shown to be NP-complete). Hence, we should not expect an algorithm for $\mathsf{PermPerm}(M)$ that is fast for any matrix M. Fortunately, powerful necessary conditions are known that may make an exhaustive search feasible.

From the practical point of view, the perm-perm symmetry of a matrix can be computed with a partition based backtracking search in a suitable permutation group as described in Leon (1991). There is also a highly optimized implementation of this search method programming language C (available from Leon via his homepage). In addition, the authors implemented a search in the language GAP. Both implementations are available in the library AREP (see Section 5), and are able to handle 100×100 matrices stemming from signal transforms in seconds.

The perm-perm symmetry of a matrix is of interest beyond the application to signal processing we have in mind here. Given an incidence structure (V, B, I) (so V and B are disjoint finite sets and $I \subseteq V \times B$) one can obtain the automorphism group of (V, B, I) as the perm-perm symmetry of its incidence matrix (the matrix $M \in \{0,1\}^{V \times B}$ such that $M_{v,b} = 1$ if and only if $(v, b) \in I$). As an example, the vertices of a graph are incident to the edges. This way, the perm-perm symmetry is closely related to automorphism groups of many discrete structures.

3.3. Mon-Mon Symmetry

The *mon-mon* symmetry of a matrix M is a generalization of the perm-perm symmetry. Let $\mathrm{Mon}_n(\mathbb{F})$ denote the group of monomial $(n \times n)$-matrices with entries from the field $\mathbb{F}$. Then we define for the matrix $M \in \mathbb{F}^{n \times m}$

$$\mathsf{MonMon}(M) = \{(L, R) \in \mathrm{Mon}_n(\mathbb{F}) \times \mathrm{Mon}_m(\mathbb{F}) \mid LM = MR\}.$$

It is easy to check that $\mathsf{MonMon}(M)$ is a group. Unfortunately, for infinite $\mathbb{F}$, the group $\mathsf{MonMon}(M)$ is not finite since scalars can be moved freely from the left of M to the right. This "scalar symmetry" conveys no structural information about M, i.e., the "interesting" part of the symmetry is given by the factor group

$$\mathsf{MonMon}(M)/\mathbb{F}^\times = \mathsf{MonMon}(M)/\{(x \cdot \mathbf{1}_n, x \cdot \mathbf{1}_m) \mid x \neq 0, x \in \mathbb{F}\}.$$

To obtain a tractable search problem, we restrict ourselves to a subtype of the mon-mon symmetry, which considers only the finite group of all monomial matrices with kth roots of unity as non-zero entries. We call these matrices *k-monomial* and denote the group of all k-monomial matrices of size $n \times n$ by $\mathrm{Mon}_n(\mathbb{F}, k)$. The parameter k is fixed and chosen depending on the given matrix M. Formally, we want to find the mon-mon symmetry *of* *order* k, defined by

$$\mathsf{MonMon}_k(M) = \{(L,R) \in \mathrm{Mon}_n(\mathbb{F},k) \times \mathrm{Mon}_m(\mathbb{F},k) \mid LM = MR\}.$$

Briefly summarized, our approach computes the mon-mon symmetry of order k of $M \in \mathbb{F}^{n\times m}$ by computing the perm-perm symmetry of the larger matrix $\mathcal{C}_k(M) \in \mathbb{F}^{kn\times km}$, where $\mathcal{C}_k$ is a suitable *coding* function. In other words, we compute $\mathsf{MonMon}_k(M)$ via $\mathsf{PermPerm}(\mathcal{C}_k(M))$. The idea for this approach is based on a method described in Leon (1991) for finding the mon-mon symmetry over finite fields. We detail the approach in the following, starting with defining the coding function $\mathcal{C}_k$.

Definition 3.3 Let $x \in \mathbb{F}$ and $k \geq 1$. We assume that the characteristic of $\mathbb{F}$ is zero or does not divide k and let ω_k denote a primitive kth root of unity in $\mathbb{F}$. We call

$$\mathcal{C}_k(x) = \begin{bmatrix} x\cdot\omega_k^0 & x\cdot\omega_k^1 & \cdots & x\cdot\omega_k^{(k-1)} \\ x\cdot\omega_k^1 & x\cdot\omega_k^2 & \cdots & x\cdot\omega_k^0 \\ \vdots & \vdots & \ddots & \vdots \\ x\cdot\omega_k^{(k-1)} & x\cdot\omega_k^0 & \cdots & x\cdot\omega_k^{(k-2)} \end{bmatrix} = [x\cdot\omega_k^{i+j} \mid 0 \leq i,j \leq k-1]$$

the *k-coding of x*. For $M \in \mathbb{F}^{n\times m}$ we analogously call

$$\mathcal{C}_k(M) = [\mathcal{C}_k(M_{i,j}) \mid 1 \leq i \leq n, 1 \leq j \leq m] \in \mathbb{F}^{kn\times km}$$

the *k-coding of M*.

The key property of $\mathcal{C}_k$ is that, for $x \in \mathbb{F}$, the $(i+1)$th row of $\mathcal{C}_k(x)$ is obtained from the ith row by multiplication with ω_k, for $1 \leq i \leq n-1$; the first row is obtained from the nth row this way. An analogous property holds for the columns and we get the following lemma.

Lemma 3.4

$$\mathcal{C}_k(\omega_k^\ell \cdot x) = (1,\ldots,k)^\ell \cdot \mathcal{C}_k(x) = \mathcal{C}_k(x)\cdot(1,\ldots,k)^{-\ell}.$$

Corresponding to the coding function $\mathcal{C}_k$, we define a group homomorphism $\mathcal{P}_k$ that embeds $\mathrm{Mon}_n(\mathbb{F},k)$ into S_{kn}. Before we state the general definition we give an illustrative example. We consider $k = 3$ and the 3-monomial matrix $S = [(1,2),(1,\omega_3^2)] = (1,2)\cdot \mathrm{diag}(1,\omega_3^2)$. The permutation matrix $\mathcal{P}_k(S)$ is obtained by replacing each entry in S by a (3×3)-matrix: zero entries are replaced by the all-zero matrix, and entries ω_3^i are replaced by $(1,2,3)^i$. We visualize this by emphasizing the resulting block structure:

$$S = \begin{bmatrix} 0 & \omega_3^2 \\ 1 & 0 \end{bmatrix} \leftrightarrow \mathcal{P}_k(S) = \left[\begin{array}{ccc|ccc} 0&0&0&0&0&1 \\ 0&0&0&1&0&0 \\ 0&0&0&0&1&0 \\ \hline 1&0&0&0&0&0 \\ 0&1&0&0&0&0 \\ 0&0&1&0&0&0 \end{array}\right].$$

The block structure of $\mathcal{P}_k(S)$ can be expressed by the following decomposition.

$$S = [(1,2),(1,\omega_3^2)] \leftrightarrow \mathcal{P}_k(S) = ((1,2)\otimes \mathbf{1}_3)\cdot\left((1,2,3)^0 \oplus (1,2,3)^2\right).$$

For general k and n, the block permutations in S_{kn} arising this way have the structure

$$(\sigma \otimes \mathbf{1}_k)\cdot(\tau_1 \oplus \ldots \oplus \tau_n),$$

where $\sigma \in \mathsf{S}_n$ represents the "macro" permutation and the τ_i are powers of the k-cycle $(1, \ldots, k)$. The group of all these permutations is the *wreath product* $\mathsf{Z}_k \wr \mathsf{S}_n$ (James and Kerber, 1981) in its natural permutation representation on kn points.

Definition 3.5 Let $\mathbb{F}$ and k be as in Definition 3.3 and let $0 \leq u_i \leq k-1$, $1 \leq i \leq n$. We define the mapping $\mathcal{P}_k$ by

$$\begin{aligned} \mathcal{P}_k : \quad & \mathrm{Mon}_n(\mathbb{F}, k) \to \mathsf{Z}_k \wr \mathsf{S}_n \text{ (as a subgroup of } \mathsf{S}_{kn}), \\ & [\sigma, (\omega_k^{u_1}, \ldots, \omega_k^{u_n})] \mapsto (\sigma \otimes \mathbf{1}_k) \cdot \textstyle\bigoplus_{i=1}^{n} (1, \ldots, k)^{u_i}. \end{aligned}$$

By construction, it is clear that $\mathcal{P}_k$ is a group isomorphism.

Our algorithm for finding the mon-mon symmetry of order k for a given matrix M is based on the following theorem. It shows that the mon-mon symmetry of order k is contained, via the mapping $\mathcal{P}_k$, in the perm-perm symmetry of $\mathcal{C}_k(M)$. The proof is straightforward using the definitions of $\mathcal{C}_k$ and $\mathcal{P}_k$ and Lemma 3.4.

Theorem 3.6 *Let $x \in \mathbb{F}$ and $k \geq 1$. We assume that the characteristic of $\mathbb{F}$ is zero or does not divide k. Let $M \in \mathbb{F}^{n \times m}$. Then*

$$(L, R) \in \mathsf{MonMon}_k(M) \Rightarrow (\mathcal{P}_k(L), \mathcal{P}_k(R)) \in \mathsf{PermPerm}(\mathcal{C}_k(M)).$$

We note that the converse is not true in general, i.e., to "decode" the mon-mon symmetry of order k of M, we need to first intersect the perm-perm symmetry of $\mathcal{C}_k(M)$ with $(\mathsf{Z}_k \wr \mathsf{S}_n) \times (\mathsf{Z}_k \wr \mathsf{S}_n)$. For all our practical applications, however, it has turned out that the intersection is not necessary, i.e., $\mathsf{PermPerm}(\mathcal{C}_k(M)) \cong \mathsf{MonMon}_k(M)$.

The remaining question is the appropriate choice of the parameter k. If $\mathbb{F} = \mathbb{F}_q$ is finite with q elements, $k = q - 1$ can be chosen, which guarantees that the entire mon-mon symmetry is found. For $\mathbb{F} \leq \mathbb{C}$ a matrix can have mon-mon symmetries of arbitrary order k. For example,

$$M = \begin{bmatrix} a & 0 \\ 0 & b \end{bmatrix}, \quad a, b \in \mathbb{F},$$

has the mon-mon symmetry

$$\begin{bmatrix} \omega_k & 0 \\ 0 & \omega_k^{-1} \end{bmatrix} \cdot M = M \cdot \begin{bmatrix} \omega_k & 0 \\ 0 & \omega_k^{-1} \end{bmatrix}, \quad \text{for all } k = 1, 2, \ldots .$$

If a matrix with complex entries has a monomial symmetry of order k, then the symmetry permutes entries with equal absolute value. For this reason, we consider all quotients $M_{i,j}/M_{k,\ell}$ with $|M_{i,j}| = |M_{k,\ell}| \neq 0$. These quotients are roots of unity and we choose k as the least common multiple of the order of those roots for which the order is finite.

As a summary, we give in Figure 2 pseudocode for computing the mon-mon symmetry of order k for a matrix M.

Regarding the computational complexity, finding mon-mon symmetry is not easier than finding perm-perm symmetry. Moreover, the encoding method described above increases the size of the matrix from $n \times m$ to $kn \times km$ where the parameter k depends on the base field over which to search for symmetries. The availability of very fast implementations for the perm-perm symmetry (see Section 3.2) makes our approach a viable solution for small k. For real matrices, and hence for most signal transforms, it is sufficient to choose $k = 2$ as the only real roots of unity are $\{-1, 1\}$. We refer the reader to Section 6 (in particular 6.4) for runtime examples.

$\mathsf{MonMon}_k(M)$:
choose the parameter k
compute the k-coding $\mathcal{C}_k(M)$
compute $P := \mathsf{PermPerm}(\mathcal{C}_k(M))$
intersect (if necessary) $P' := P \cap (\mathsf{Z}_k \wr \mathsf{S}_n) \times (\mathsf{Z}_k \wr \mathsf{S}_n)$
return $\mathsf{MonMon}_k(M) = \{(\mathcal{P}_k^{-1}(L), \mathcal{P}_k^{-1}(R)) \mid (L, R) \in P'\}$

Figure 2. The algorithm for finding $\mathsf{MonMon}_k(M)$ for a matrix M including the choice of k.

3.4. Perm-Mat Symmetry

As a building block to compute the perm-irred and the mon-irred symmetry, we now consider the *perm-mat* symmetry. As the name indicates, we are looking for all pairs (L, R) where L is a permutation matrix, R can be any matrix, and $LM = MR$. Formally, for $M \in \mathbb{F}^{n\times m}$, we define

$$\mathsf{PermMat}(M) = \{(L, R) \in \mathsf{S}_n \times \mathrm{GL}_m(\mathbb{F}) \mid LM = MR\}.$$

Not all cases of the perm-mat symmetry are interesting. For example, if M is invertible then there is a matrix R, namely $R = M^{-1}LM$, for any permutation L, in which case $\mathsf{PermMat}(M)$ is just (isomorphic to) the symmetric group S_n. In addition, as explained in Section 3.1, the subdirect structure allows to eliminate identical copies of rows (the perm part) and to eliminate linear dependent rows (the mat part). The following lemma is the basis for computing the identified factor group of the subdirect product for the perm-mat symmetry for a matrix M, which has more rows than columns. For notational compactness we use the notation $M_{I,J}$ to indicate submatrices of M, where I and J are either integers, sets or lists of integers, or the symbol "$*$" denoting the full index set. For example, $M_{i,*}$ denotes the i-th row of M, $M_{*,j}$ denotes the j-th column, and $M_{I,*}$ denotes the submatrix of rows of M with index $i \in I$.

Lemma 3.7 *Let $M \in \mathbb{F}^{n\times m}$ be a matrix with $n \geq m$ and assume that the rows of M are pairwise distinct, and that the columns of M are linearly independent. Choose an m-tuple I of row indices such that the submatrix $M_{I,*}$ is invertible. Let $L(I)$ denote the image of I under the permutation L. Then*

$$\mathsf{PermMat}(M) = \{(L, M_{I,*}^{-1} M_{L(I),*}) \mid L \cdot M = M \cdot M_{I,*}^{-1} M_{L(I),*}\}.$$

Proof. If $LM = MM_{I,*}^{-1}M_{L(I),*}$ for some permutation L then $(L, M_{I,*}^{-1}M_{L(I),*}) \in \mathsf{PermMat}(M)$. Conversely, let $(L, R) \in \mathsf{PermMat}(M)$ and consider the rows in I:

$$(LM)_{I,*} = M_{L(I),*} = M_{I,*}R = (MR)_{I,*} \Rightarrow R = M_{I,*}^{-1}M_{L(I),*},$$

as desired. $\square$

The preceeding lemma states that a permutation L of the perm-mat symmetry is already defined by its image $L(I)$ on a certain base I of size m. Moreover, the mapping $L \mapsto M_{I,*}^{-1}M_{L(I),*}$ is an isomorphism from the permutation group on the left of M to the corresponding matrix group on the right.

```
PermMat(G, M):
    choose a list I of row indices such that M_{I,*} is invertible; — (1)
    precompute M · (M_{I,*})^{-1}, {M_{k,ℓ} | k, ℓ} and {M_{k,*} | k};
    H := E (the trivial subgroup of G);
    for J ∈ orbit of I under G do
        L := findperm(M, I, J);
        if L ≠ false and L ∈ G then
            extend H by L
    return H.

findperm(M, I, J):
    L := id; (the identity mapping on {1, ..., n})
    for i ∈ {1, ..., n} do
        r := [ ];
        for j ∈ {1, ..., m} do
            r[j] := (M · (M_{I,*})^{-1})_{i,1} · M_{J[1],j} + ··· + (M · (M_{I,*})^{-1})_{i,m} · M_{J[m],j};
            if r[j] ∉ {M_{k,ℓ} | k, ℓ} then return false;
        if r ∉ {M_{k,*} | k} then return false;
        L(i) := (k such that r = M_{k,*}); — (2)
    if not L is a permutation return false;
    return L.
```

Figure 3. $\mathsf{PermMat}(M)$ for an $(n \times m)$-matrix M with linear independent columns and distinct rows. The argument G allows to restrict the search to a subgroup of the symmetric group.

The lemma is the basis for the correctness of the algorithm shown in pseudocode in Figure 3. It computes

$$\mathsf{PermMat}(G, M) = \{L \mid (L, R) \in \mathsf{PermMat}(M), L \in G\},$$

for an $(n \times m)$-matrix M with linear independent columns and pairwise unequal rows and a subgroup G of the full permutation group S_n. The additional argument G is useful for the application of the algorithm to computing the perm-irred symmetry. Note that the list I in statement (1) exists because the columns of M are linear independent and that the image $L(i)$ in statement (2) is uniquely defined because the rows of M are distinct. The algorithm uses the local function $\mathsf{findperm}$ to test if $\tilde{M} = M M_{I,*}^{-1} M_{J,*}$ is a row-permuted version of M, and, if so, to compute the permutation. (The variable r contains the i-th row of $\tilde{M}$.) The function interleaves constructing $\tilde{M}$ and testing its properties to allow an early return. $\mathsf{PermMat}$ may invoke $\mathsf{findperm}$ up to $m!\binom{n}{m}$ times.

3.5. Perm-Irred Symmetry

This section explains how to find *perm-irred* symmetries of a matrix. Throughout this section we assume the matrix M to be square of size $n \times n$ and invertible. Following the mnemonic names of Table 1, a perm-irred symmetry of M is a pair of representations (ϕ, ϕ^M) such that ϕ is a permutation representation and ϕ^M is a permuted direct sum of irreducible representations. In other words, there is a permutation $\pi \in \mathsf{S}_n$ such that

$$\phi^M = (\phi_1 \oplus \cdots \oplus \phi_r)^\pi, \quad \text{where all } \phi_i \text{ are irreducible.}$$

Unlike for the perm-perm symmetry, there is no largest perm-irred symmetry containing all others. Therefore we postpone the formal definition of the search problem and start by defining a quantitative measure of block structure.

Definition 3.8 Let A be a square matrix (not necessarily invertible) of size n. The *conjugated block structure* (cbs) of A is the partition

$$\operatorname{cbs}(A) = \{1, \ldots, n\}/\sim^*,$$

where $\sim^*$ is the reflexive-symmetric-transitive closure of the binary relation $\sim$ defined on $\{1, \ldots, n\}$ by $i \sim j \Leftrightarrow A_{ij} \neq 0$.

For the following investigations we introduce additional notation. Let $\sqsubseteq$ denote the partial order defined on partitions of $\{1, \ldots, n\}$ (read $p \sqsubseteq q$ as "p refines q") and define $\sqcap$ ("meet": coarsest common refinement) and $\sqcup$ ("join": finest common union of blocks) as the lattice operations associated with the refinement relation. Moreover, let p^π denote the partition obtained from p by renumbering the points with permutation π. Finally, let $p \oplus q$ denote the partition of $\{1, \ldots, n+m\}$ obtained by concatenating the blocks of p partitioning n points and q partitioning m points, formally $p \oplus q = p \cup \{n + b \mid b \in q\}$.

The purpose of the cbs is to indicate how far a matrix A decomposes into a direct sum $A_1 \oplus \cdots \oplus A_r$ if the rows and columns are renumbered properly by conjugating with a permutation. We will use the following properties of the cbs.

Lemma 3.9 *For square matrices A, B and a permutation π,*

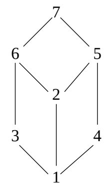
(i) $\operatorname{cbs}(A^\pi) = \operatorname{cbs}(A)^\pi$.
(ii) $\operatorname{cbs}(A \oplus B) = \operatorname{cbs}(A) \oplus \operatorname{cbs}(B)$.
(iii) $\operatorname{cbs}(AB) \sqsubseteq \operatorname{cbs}(A) \sqcup \operatorname{cbs}(B)$ *if A and B are of the same size.*
(iv) $\operatorname{cbs}(A^{-1}) = \operatorname{cbs}(A)$ *if A is invertible.*

Proof. We prove the properties one by one.

(i) Compatibility with conjugation follows from $A_{ij} = (A^\pi)_{\pi(i),\pi(j)}$ for all i, j.
(ii) Compatibility with the direct sum is based on the fact that the relation $i \sim j \Leftrightarrow (A \oplus B)_{ij} \neq 0$ already partitions the set $\{1, \ldots, n\}$ into two unconnected subsets, and taking the reflexive-symmetric-transitive closure does not merge unconnected subsets.
(iii) Compatibility with matrix multiplication is a consequence of the first two properties.
(iv) Let $A = (A_1 \oplus \cdots \oplus A_r)^\pi$ be a finest decomposition of A into a permuted direct sum. Since matrix inversion is compatible with conjugation and with direct sum, this implies $A^{-1} = (A_1^{-1} \oplus \cdots \oplus A_r^{-1})^\pi$. As the decomposition of A was assumed finest, A_i cannot be decomposed further and $\operatorname{cbs}(A_i)$ is the coarsest partition for all i. Hence, $\operatorname{cbs}(A^{-1})$ is a refinement of $\operatorname{cbs}(A)$ and equality follows from applying this twice as $\operatorname{cbs}(A) \sqsubseteq \operatorname{cbs}(A^{-1}) \sqsubseteq \operatorname{cbs}(A)$.

□

Table 2. The lattice of all block structures found in $\mathsf{S}_6^{\mathrm{DFT}_6}$.



	p	$\Gamma(p)$	$\lvert\Gamma(p)\rvert$	generators
1	$(1\|2\|3\|4\|5\|6)$	Z_6	6	$(1,2,3,4,5,6)$
2	$(1\|2\ 6\|3\ 5\|4)$	D_{12}	12	$(2,6)(3,5),(1,2,3,4,5,6)$
3	$(1\|2\ 5\|3\ 6\|4)$	$\mathsf{Z}_3\times\mathsf{S}_3$	18	$(1,3,5)(2,4,6);(1,2)(3,4)(5,6),(1,3,5)(2,6,4)$
4	$(1\|2\ 4\ 6\|3\|5)$	$\mathsf{Z}_2\times\mathsf{A}_4$	24	$(1,4)(2,5)(3,6);(1,2,3)(4,5,6),(1,2,6)(3,4,5)$
5	$(1\|2\ 4\ 6\|3\ 5)$	$\mathsf{Z}_2\times\mathsf{S}_4$	48	$(1,4)(2,5)(3,6);(2,3)(5,6),(1,2,4,5)(3,6)$
6	$(1\|2\ 3\ 5\ 6\|4)$	$\mathsf{S}_3\wr\mathsf{Z}_2$	72	$(1,2)(3,4)(5,6),(1,3,5)(2,6,4);(4,6)$
7	$(1\|2\ 3\ 4\ 5\ 6)$	S_6	720	$(1,2),(1,2,3,4,5,6)$

We illustrate the cbs and Lemma 3.9 (i) with the following example (dots represent entries of zero):

$$\mathrm{cbs}\left(\begin{bmatrix}1&\cdot&2&\cdot&\cdot\\ \cdot&3&\cdot&\cdot&4\\ 5&\cdot&6&\cdot&\cdot\\ \cdot&\cdot&\cdot&7&\cdot\\ \cdot&8&\cdot&\cdot&9\end{bmatrix}\right)=\mathrm{cbs}\left(\begin{bmatrix}1&2&\cdot&\cdot&\cdot\\ 5&6&\cdot&\cdot&\cdot\\ \cdot&\cdot&3&4&\cdot\\ \cdot&\cdot&8&9&\cdot\\ \cdot&\cdot&\cdot&\cdot&7\end{bmatrix}^{(2,3)(4,5)}\right)=(12|34|5)^{(2,3)(4,5)}=(13|25|4).$$

Now we use the cbs to find the perm-irred symmetry. Given a matrix M, we can relate permutation groups $G\le\mathsf{S}_n$ and block structures $p\sqsubseteq\{\{1,\dots,n\}\}$ by the mappings Π and Γ defined by

$$\begin{aligned}\Pi(G)&=\bigsqcup_{L\in G}\mathrm{cbs}(M^{-1}LM),\quad\text{and}\\ \Gamma(p)&=\{L\in\mathsf{S}_n\mid\mathrm{cbs}(M^{-1}LM)\sqsubseteq p\}.\end{aligned}$$

This means, $\Pi(G)$ is the block structure that the group G admits under conjugation with the matrix M and $\Gamma(p)$ is the largest group $G\le\mathsf{S}_n$ admitting the block structure p. Note that $\Pi(G)$ can be found by computing $\mathrm{cbs}(M^{-1}LM)$ for a generating set of G and using Lemma 3.9.

Lemma 3.10 Π *and* Γ *are order preserving mappings between the lattice of subgroups* G *of* S_n *and the lattice of partitions* p *of* $\{1,\dots,n\}$. *Moreover,*

$$\Pi(\Gamma(p))\sqsubseteq p\quad\textit{and}\quad\Gamma(\Pi(G))\ge G\quad\textit{for all } p \textit{ and } G.$$

Proof. Consequence of the fact that cbs is compatible with matrix multiplication (Lemma 3.9) and of some simple properties of finite lattices. □

Despite the previous lemma, Π and Γ are in general no lattice homomorphisms. For example, Table 2 shows all block structures obtainable as $\mathrm{cbs}(\mathrm{DFT}_6^{-1}\,L\,\mathrm{DFT}_6)$ for permutations $L\in\mathsf{S}_6$. Yet, the partition $(1|2\,6|3|4|5)$, the "meet" of entries no. 4 and 6, is not in the table. Also, the group $\langle(\mathsf{Z}_3\times\mathsf{S}_3)\cup\mathsf{D}_{12}\rangle$, the "join" of entries no. 2 and 3, is not in the table. This shows that the lattice is neither a sublattice of all partitions of $\{1,\dots,n\}$ nor of the lattice of subgroups of S_n.

Based on Π and Γ we can now formulate the search problem for the perm-irred symmetry. Finding all perm-irred symmetries of M is done by first determining all block structures found in S_n^M and second determining all groups $G\le\mathsf{S}_n$ for which the blocks

```
PermBlock(M):
    T := ∅;
    for L ∈ S_n do
        p := cbs(M^{-1} L M);
        for G ∈ T do
            if Π(G) ⊒ p then
                extend G by L;
        if p ∉ {Π(G) | G ∈ T} then
            H := ⟨G | Π(G) ⊑ p⟩;
            extend H by L;
            insert H into T;
    return T.
```

```
PermBlock(M, k):
    T := {S_n};
    while ∃G ∈ T, b ∈ Π(G) : |b| > k do
        choose such G, b;
        remove G from T;
        for b' ⊆ b, 1 ≤ |b'| ≤ k do
            H := Γ(p ⊓ {b', {1, ..., n} − b'});
            insert H into T;
    return T.

Γ(p) (for a given M):
    G := S_n;
    for b ∈ p do
        G := PermMat(G, M_{*,b});
    return G.
```

Figure 4. Computing $\mathsf{PermBlock}(M)$ by enumerating permutations (left) and by enumerating partitions consisting of blocks of size at most k (right).

of G^M are all irreducible. Formally,

$$\begin{aligned} \mathsf{PermBlock}(M) &= \{\Gamma(p) \mid p \text{ is a partition of } \{1, \ldots, n\}\}, \\ \mathsf{PermIrred}(M) &= \{G \in \mathsf{PermBlock}(M) \mid \text{blocks of } G^M \text{ irreducible}\}. \end{aligned}$$

For example, Table 2 shows all groups in $\mathsf{PermBlock}(\mathrm{DFT}_6)$, which, in this case, is equal to $\mathsf{PermIrred}(\mathrm{DFT}_6)$. This completes the definition of the perm-irred symmetry of M. We will now present two methods to compute $\mathsf{PermBlock}(M)$. Once this is found, it is easy to extract $\mathsf{PermIrred}(M)$ from it by testing if all characters of the direct summands of G^M are indeed irreducible. Both methods are described in Egner (1997).

Permutation based Search

The first method constructs $\mathsf{PermBlock}(M)$ by enumerating all permutations $L \in \mathsf{S}_n$ and maintaining a set T of permutation groups. Pseudocode for the algorithm is shown in Figure 4 (left). The correctness of the algorithm rests on the following invariant of the loop: Let $L_1, \ldots, L_k$ be all permutations encountered so far, and define associated partitions $p_i = \mathrm{cbs}(M^{-1} L_i M)$. Then T is the set of permutation groups

$$T = \{\langle L_i \mid p_i \sqsubseteq q\rangle \mid q \in \{p_i \mid i\}\}.$$

In other words, T contains exactly one group for every partition q encountered at this stage, and for each such q the group contains all permutations leading to a decomposition of M not coarser than q. Finally, when all permutations have been considered, T is equal to $\mathsf{PermBlock}(M)$.

The approach considers all $n!$ permutations and, for each such permutation L, the $(n \times n)$-matrix multiplication $M^{-1} \cdot (LM)$ has to be computed in order to find $\mathrm{cbs}(M^{-1} LM)$. (As matrices are only permuted, scalar multiplications can be precomputed in a table of size $O(n^4)$, which is a minor improvement.) In any case, enumerating permutations means exponential running time.

Fortunately, there is a better way to approach the problem of computing $\mathsf{PermBlock}(M)$ than through enumeration of permutations.

Partition based Search

$\mathsf{PermBlock}(M) = \{\Gamma(p) \mid p\}$ can also be approached by enumerating partitions instead of permutations. This approach is motivated by the observation that $\Gamma(p)$ can be computed using $\mathsf{PermMat}$ with the algorithm shown in pseudocode in Figure 4 (lower right). The correctness of the algorithm is based on the following statement, which allows to "split off a block" using $\mathsf{PermMat}$.

Lemma 3.11 *Let $p = \{b, \{1, \ldots, n\} - b\}$ be a partition with exactly two blocks. Then*

$$\Gamma(p) = \mathsf{PermMat}(\mathsf{S}_n, M_{*,b}) = \{L \mid (L, R) \in \mathsf{PermMat}(M_{*,b})\}.$$

PROOF. Consider (L, R) such that $LM_{*,b} = M_{*,b}R$. Then R operates only on columns b of M by definition. This implies that $\mathrm{cbs}(M^{-1}LM)$ refines p. Conversely, if L is a permutation such that $\mathrm{cbs}(M^{-1}LM)$ refines p, then $M^{-1}LM$ maps the vector space spanned by the columns b onto itself. Hence, there is an R such that $LM_{*,b} = M_{*,b}R$. □

The algorithm in Figure 4 to compute $\Gamma(p)$ for arbitrary p repeatedly reduces the group G by splitting off one block of p at a time. Iterative application of Lemma 3.11 shows that the final result contains exactly those permutations L for which $\mathrm{cbs}(M^{-1}LM)$ refines p. The method is best illustrated by an example. Consider the matrix ($i = \sqrt{-1}$)

$$M = \mathrm{DFT}_4 = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & i & -1 & -i \\ 1 & -1 & 1 & -1 \\ 1 & -i & -1 & i \end{bmatrix}.$$

To compute $\Gamma(1|2\,4|3)$ for M we start with the group S_4 and split off the block $\{1\}$. This does not reduce the group as the first column of M is constant and any permutation of the rows is a symmetry. Next, we split off block $\{2, 4\}$ by computing

$$\mathsf{PermMat}(\mathsf{S}_4, M_{*,\{2,4\}}) = \mathsf{PermMat}(\mathsf{S}_4, \begin{bmatrix} 1 & 1 \\ i & -i \\ -1 & -1 \\ -i & i \end{bmatrix}).$$

The columns are linearly independent and the rows distinct. We choose $I = [1, 2]$ as a base for the rows and consider all J in the orbit of I under S_4, which is

$$\{[1,2], [1,3], [1,4], [2,1], [2,3], [2,4], [3,1], [3,2], [3,4], [4,1], [4,2], [4,3]\}.$$

Consider for example, $J = [2, 3]$. In this case, $MM_{I,*}^{-1}M_{J,*}$ is a row-permuted M:

$$MM_{[1,2],*}^{-1}M_{[2,3],*} = \begin{bmatrix} i & -i \\ -1 & -1 \\ -i & i \\ 1 & 1 \end{bmatrix} = (1, 2, 3, 4) \cdot M.$$

Hence, $(1, 2, 3, 4) \in \mathsf{PermMat}(\mathsf{S}_4, M_{*,\{2,4\}})$. On the other hand, for $J = [1, 3]$ we obtain

$$MM_{[1,2],*}^{-1}M_{[1,3],*} = \begin{bmatrix} -i & i \\ i & -i \\ i & -i \\ -i & i \end{bmatrix},$$

which is not a row-permuted version of M. Hence, no permutation L for which $L(1) = 1$ and $L(2) = 3$ is in $\mathsf{PermMat}(\mathsf{S}_4, M_{*,\{2,4\}})$. Testing the other possible images J, we find

$$\mathsf{PermMat}(\mathsf{S}_4, M_{*,\{2,4\}}) = \mathsf{D}_8 = \langle (1,4)(2,3), (1,2,3,4) \rangle,$$

(D_8 denotes a dihedral group of eight elements.) The algorithm would go on splitting off the final block $\{3\}$, but for the sake of illustration we rather compute the partition

$$\Pi(\mathsf{D}_8) = \mathrm{cbs}(M^{-1}(1,4)(2,3)M) \sqcup \mathrm{cbs}(M^{-1}(1,2,3,4)M) = (1|2\,4|3) \sqcup (1|2|3|4) = (1|2\,4|3).$$

This shows that it is not necessary to split off the block $\{3\}$ as D_8 already separates it. In effect, the example shows for $M = \mathrm{DFT}_4$ that

$$\Gamma(1|2\,4|3) = \langle (1,4)(2,3), (1,2,3,4) \rangle.$$

The algorithm to compute $\Gamma(p)$ can be used to compute $\mathsf{PermBlock}(M)$ by simply computing $\Gamma(p)$ for all partitions p of $\{1, \ldots, n\}$. Unfortunately, there are exponentially many partitions. However, the method allows to restrict the search to partitions that consist of small blocks, only! This is the purpose of the algorithm in Figure 4 (upper right). It computes all groups in $\mathsf{PermBlock}(M)$ for which the block structure consists of blocks of size at most k. For signal transforms, these symmetries turn out to be most useful for obtaining sparse factorizations.

3.6. Mon-Irred Symmetry

The *mon-irred* symmetry generalizes the perm-irred symmetry in the same way as the mon-mon symmetry generalizes the perm-perm symmetry. For $M \in \mathrm{GL}_n(\mathbb{F})$ define $\mathsf{MonMat}(M)$, $\mathsf{MonBlock}(M)$, and $\mathsf{MonIrred}(M)$ as the mon-mat, mon-block, and mon-irred structures, respectively, substituting S_n in all placed by $\mathrm{Mon}_n(\mathbb{F})$ (the group of all invertible monomial matrices of size $n \times n$).

By using the subdirect structure (Lemma 3.1) we again only consider the case where no rows of M are scalar multiples of each other. Similarly to $\mathsf{MonMon}(M)$ (see Section 3.3), also $\mathsf{MonIrred}(M)$ may be infinite, and we solve it analogously by defining $\mathsf{MonBlock}_k(M)$ and $\mathsf{MonIrred}_k(M)$, which restrict the symmetry to k-monomial matrices (i.e., containing only k-th roots of unity as non-zero elements).

The groups in $\mathsf{MonBlock}_k(M)$ can be constructed in a similar way as $\mathsf{PermBlock}(M)$. One can either enumerate all possible (k-monomial) L or can recursively split off blocks and use $\mathsf{MonMat}(M_{*,J})$ to construct the largest group stabilizing the block. We do not describe the function MonMat here as it is very similar to $\mathsf{PermMat}$. The biggest difference is that MonMat has to consider k^n times as many candidates for a k-monomial matrix of size $n \times n$, as $\mathsf{PermMat}$ has to consider for permutations of the same degree. Therefore, computing the mon-irred symmetry with our methods is only feasible for very small values of the parameter k.

4. Decomposing Monomial Representations

The second important step in the matrix factorization algorithm is the decomposition of monomial representations. As explained in Section 2, we are not only interested in the irreducible components contained in a monomial representation ϕ, but also in the corresponding decomposition matrix of ϕ given as a product of structured sparse matrices.

The decomposition algorithm decomposes arbitrary monomial representations of solvable groups and is comprehensively described in Püschel (2002), which builds on ideas of Minkwitz (1995). For the sake of completeness, we briefly survey the algorithm in this section, restricting ourselves to its structure and main steps. Before we give the algorithm we restate the main results it is based on.

4.1. Background for the Algorithm

We restate the following three theorems from Püschel (2002), where they can be found as Theorem 3.16, 3.33, and 3.34, respectively.

Let ϕ be a monomial representation of a solvable group G. We recall that ϕ is called transitive, if it cannot be conjugated by a permutation to be a direct sum. The following result connects transitive monomial representations and inductions.

Theorem 4.1 *Let ϕ be a transitive monomial representation of a group G. Then there exists a diagonal matrix D, a subgroup $H \leq G$ with representation λ_H of degree one, and a transversal T of G/H such that*

$$\phi^D = \lambda_H \uparrow_T G \quad \textit{(induction of } \lambda \textit{ to } G \textit{ with transversal } T\textit{).}$$

Let $N \trianglelefteq G$ be a normal subgroup of G of prime index p and assume ϕ is a representation of N with decomposition matrix A. Theorem 4.2 explains how to construct a decomposition matrix of the induction $\phi \uparrow_T G$, and Theorem 4.3 explains how to construct a decomposition matrix for the extension $\overline{\phi}$ (if it exists). Both theorems are essentially based on Clifford's theory† (Curtis and Reiner, 1962).

These two cases constitute the core of our decomposition algorithm. For the purpose of this paper, the reader may skip the technical details; only the two formulas for the decomposition matrix B are of importance. Note that all the factors in the formulas are sparse. Finally, these formulas explain the structure of the factorizations we will present in Section 6.

Theorem 4.2 *Let $N \trianglelefteq G$ be a normal subgroup of prime index p and T a transversal of G/N. Assume ϕ is a representation of N of degree n with decomposition matrix A such that $\phi^A = \bigoplus_{i=1}^k \rho_i$, where $\rho_1, \ldots, \rho_j$ are exactly those among the ρ_i having an extension $\overline{\rho}_i$ to G. Denote by $d = \deg(\rho_1) + \ldots + \deg(\rho_j)$ the entire degree of the extensible ρ_i and set $\overline{\rho} = \overline{\rho}_1 \oplus \ldots \oplus \overline{\rho}_j$. Then there exists a permutation matrix P such that*

$$B = (\mathbf{1}_p \otimes A) \cdot P \cdot \left(\bigoplus_{t \in T} \overline{\rho}(t) \oplus \mathbf{1}_{p(n-d)} \right) \cdot \left((\mathrm{DFT}_p \otimes \mathbf{1}_d) \oplus \mathbf{1}_{p(n-d)} \right)$$

is a decomposition matrix of $\phi \uparrow_T G$.

Theorem 4.3 *Let $N \trianglelefteq G$ be a normal subgroup of prime index p with transversal $T = (t^0, t^1, \ldots, t^{p-1})$ and representation ϕ a over the field $\mathbb{F}$. Assume that ϕ has an extension $\overline{\phi}$ to G. Further let A decompose ϕ such that equivalent irreducibles are equal and adjacent, $\phi^A = \bigoplus_{i=1}^k R_i$, where $R_i = \rho_i^{n_i}$ is a homogeneous component of multiplicity n_i. We*

† The two theorems do *not* correspond to the induction case and the extension case of Clifford's theory; both theorems need both cases.

```
MonDec(φ):
    case: φ irreducible
        return 1_deg(φ)
    case: φ intransitive
        decompose φ^P = φ_1 ⊕ ... ⊕ φ_k,   φ_i transitive, P permutation
        return P · (MonDec(φ_1) ⊕ ... ⊕ MonDec(φ_k))
    case: φ transitive and not an induction
        decompose φ^D = λ_H ↑_T G,   D is diagonal (Theorem 4.1)
        return D · MonDec(λ_H ↑_T G)
    case: φ = λ_H ↑_T G and exists normal subgroup N with H ≤ N ⊴ G, |G/N| prime
        decompose (λ_H ↑_T G)^M = (λ_H ↑_{T_1} N) ↑_{T_2} G,   M is monomial
        A := MonDec(λ_H ↑_{T_1} N)
        return M · B, where B is computed with Theorem 4.2
    case: φ = λ_H ↑_T G
        A := MonDec((λ_H ↑_T G) ↓ N)
        return B, computed with Theorem 4.3
```

Figure 5. The divide-and-conquer algorithm (sketched) for computing a factorized, structured decomposition matrix for a monomial representation ϕ of a solvable group G.

denote $d_i = \deg(\rho_i)$. Furthermore, we require that whenever $R_i \cong R_j^{t^\ell}$ then even $R_i = R_j^{t^\ell}$ and that these components are adjacent, ordered according to $R_i, R_i^t, \ldots, R_i^{t^{p-1}}$. Then there exist invertible matrices $A_i \in \mathbb{F}^{n_i \times n_i}$ and a permutation matrix P such that

$$B = A \cdot \left(\bigoplus_{i=1}^{k} A_i \otimes \mathbf{1}_{d_i} \right) \cdot P$$

is a decomposition matrix of the extension $\overline{\phi}$.

4.2. The Decomposition Algorithm

Figure 5 shows pseudocode for the decomposition algorithm, which uses a "divide-and-conquer" approach that recurses over the structure of ϕ by repeatedly considering a cascade of different cases. We concentrate on the computation of the decomposition matrix only, omitting the computation of the corresponding irreducible components of ϕ. In the actual algorithm, both computations are necessarily intertwined (as can be seen from Theorems 4.2 and 4.3).

We give an overview of the algorithm. If ϕ is irreducible, the identity is a decomposition matrix. If ϕ is not transitive, we decompose it with a permutation into a direct sum of transitive monomial representations, which are decomposed recursively. If ϕ is transitive, we apply Theorem 4.1 to reduce the decomposition problem to the case of an induction $\lambda_H \uparrow_T G$. Since G is solvable, we now find a normal subgroup $N \trianglelefteq G$ of prime index and either $H \leq N$ or $H \not\leq N$. If $H \leq N$ then we decompose $\lambda_H \uparrow_T G$ with a monomial matrix M into a double induction $(\lambda_H \uparrow_T G)^M = (\lambda_H \uparrow_{T_1} N) \uparrow_{T_2} G$ (explained in Püschel (2002)) and recurse with the lower induction to find a decomposition matrix A. The conquer step finds the decomposition matrix B using Theorem 4.2. In the other case, $H \not\leq N$, we recurse with the restriction $(\lambda_H \uparrow_T G) \downarrow N$. The conquer step is solved by Theorem 4.3. Since both cases reduces the size of the group represented, the algorithm terminates.

5. The Library AREP

The authors have implemented the methods described in this article in the software library AREP (Egner and Püschel, 1998), a refereed shared package written in the computer algebra language GAP (1997). The two central data types are `AMat` (Abstract Matrix) and `ARep` (Abstract Representation), which are recursive data structures to efficiently represent structured matrices (like direct sum or tensor product) and structured representations (like induction or conjugation) in a symbolic form. Based on these data types AREP contains

1. functions to efficiently manipulate and compute with structured matrices and representations;
2. functions to find the different types of symmetry described in Section 3;
3. a function to decompose monomial representations of solvable groups into irreducibles as sketched in Section 4; and, combining 2. and 3.,
4. functions to construct sparse factorizations for a given matrix as described in Section 2.

Furthermore, AREP is interfaced with the SPIRAL-system (Moura *et al.*, 1998), which allows the user to generate C or Fortran code for each fast algorithm found by AREP (Egner *et al.*, 2001). For more information on AREP we refer the reader to the website of AREP (Egner and Püschel, 1998).

6. Examples

This section is a gallery of a few examples for the matrix factorization algorithm presented in this article. The considered matrices are discrete signal transforms following the definitions of Elliott and Rao (1982) and Rao and Yip (1990). All factorizations (i.e., fast transform algorithms) have been generated verbatim as they are presented (even in the LaTeX format). We state the symmetries found (with varying detail) and the runtime needed to generate the sparse factorization. All experiments where run on an Athlon 1100 MHz, running Linux. For a comparison of our generated fast transform algorithms to the algorithms known from the literature we refer the reader to Egner and Püschel (2001).

In addition to the notation introduced in Section 1.3 we will use

$$\mathrm{R}_\alpha = \begin{bmatrix} \cos(\alpha) & \sin(\alpha) \\ -\sin(\alpha) & \cos(\alpha) \end{bmatrix},$$

to denote a (2×2)-rotation matrix with angle α, and

$$\mathrm{DFT}_2 = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}.$$

for the discrete Fourier transform of size 2×2.

6.1. Discrete Fourier Transform

The discrete Fourier transform of size n is defined by the matrix

$$\mathrm{DFT}_n = \left[\omega_n^{kl} \mid 0 \leq k, \ell < n\right].$$

As it is well-known, the DFT_n has a perm-irred symmetry (ϕ, ψ) with cyclic symmetry group $\mathsf{Z_n} = \langle x \mid x^n = 1\rangle$,

$$\phi: \ x \mapsto [(1, 2, \ldots, n), n], \quad \psi: \ x \mapsto \mathrm{diag}(\omega_n^0, \omega_n^1, \ldots, \omega_n^{n-1}).$$

As an example we consider $n = 8$. We find this perm-irred symmetry, and, based on it, the Cooley-Tukey fast Fourier transform (FFT) algorithm,

$$\begin{aligned}\mathrm{DFT}_8 &= (\mathrm{DFT}_2 \otimes \mathbf{1}_4) \cdot \mathrm{diag}(1,1,1,1,1,\omega_8,\omega_4,\omega_8^3) \\ &\cdot(\mathbf{1}_2 \otimes \mathrm{DFT}_2 \otimes \mathbf{1}_2) \cdot \mathrm{diag}(1,1,1,\omega_4,1,1,1,\omega_4) \\ &\cdot(\mathbf{1}_4 \otimes \mathrm{DFT}_2) \cdot [(2,5)(4,7),8].\end{aligned}$$

This factorization has been generated in 1.4 seconds.

6.2. Discrete Cosine Transform, type II and III

The (unscaled) discrete cosine transform of type III is defined by the matrix

$$\mathrm{DCT\text{-}III}_n = [\cos((2k+1)\ell\pi/2n) \mid 0 \le k, \ell < n].$$

For $n = 8$ we find a perm-irred symmetry (ϕ, ψ) with dihedral symmetry group $\mathsf{D}_{16} = \langle x, y \mid x^8 = y^2 = 1, x^y = x^{-1} \rangle$,

$$\begin{array}{ll}\phi: \ x \mapsto [(1,3,5,7,8,6,4,2),8], & y \mapsto [(2,3)(4,5)(6,7),8], \\ \psi: \ x \mapsto M_1, & y \mapsto M_2.\end{array}$$

Using the shorthand notation $c_k = \cos(k\pi/8)$ and $s_k = \sin(k\pi/8)$, the matrices M_1 and M_2 are given by

$$M_1 = \begin{bmatrix} c_0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & c_2 & 0 & 0 & 0 & 0 & 0 & s_2 \\ 0 & 0 & c_4 & 0 & 0 & 0 & s_4 & 0 \\ 0 & 0 & 0 & c_6 & 0 & s_6 & 0 & 0 \\ 0 & 0 & 0 & 0 & c_8 & 0 & 0 & 0 \\ 0 & 0 & 0 & s_{10} & 0 & c_{10} & 0 & 0 \\ 0 & 0 & s_{12} & 0 & 0 & 0 & c_{12} & 0 \\ 0 & s_{14} & 0 & 0 & 0 & 0 & 0 & c_{14} \end{bmatrix}, \quad M_2 = \begin{bmatrix} c_0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & c_1 & 0 & 0 & 0 & 0 & 0 & s_1 \\ 0 & 0 & c_2 & 0 & 0 & 0 & s_2 & 0 \\ 0 & 0 & 0 & c_3 & 0 & s_3 & 0 & 0 \\ 0 & 0 & 0 & 0 & s_4 & 0 & 0 & 0 \\ 0 & 0 & 0 & s_5 & 0 & c_5 & 0 & 0 \\ 0 & 0 & s_6 & 0 & 0 & 0 & c_6 & 0 \\ 0 & s_7 & 0 & 0 & 0 & 0 & 0 & c_7 \end{bmatrix}.$$

The representation ψ is a permuted direct sum of irreducible representations of degrees 1 and 2: $\mathrm{cbs}(M_1) = \mathrm{cbs}(M_2) = (1|2\,8|3\,7|4\,6|5)$.

Based on this symmetry we find the factorization

$$\begin{aligned}\mathrm{DCT\text{-}III}_8 &= [(1,2,6,8)(3,7,5,4),8] \cdot (\mathbf{1}_2 \otimes ((\mathbf{1}_2 \otimes \mathrm{DFT}_2) \cdot [(2,3),4] \cdot (\mathrm{DFT}_2 \oplus \mathbf{1}_2))) \\ &\cdot[(2,7,6,8,5,4,3),8] \cdot (\mathbf{1}_4 \oplus \tfrac{1}{\sqrt{2}} \cdot \mathrm{DFT}_2 \oplus \mathbf{1}_2) \cdot [(5,6),8] \\ &\cdot((\mathrm{DFT}_2 \otimes \mathbf{1}_3) \oplus \mathbf{1}_2) \cdot [(2,8,3,7,4),8] \\ &\cdot(\mathrm{diag}(1,\tfrac{1}{\sqrt{2}}) \oplus \mathrm{R}_{\frac{13}{8}\pi} \oplus \mathrm{R}_{\frac{17}{16}\pi} \oplus \mathrm{R}_{\frac{11}{16}\pi}) \cdot [(2,5)(4,7)(6,8),8]\end{aligned}$$

The factorization was generated in 1.9 seconds. The discrete cosine transform of type II, DCT-II, is the transpose of DCT-III and we obtain a factorization for $\mathrm{DCT\text{-}II}_8$ by symbolic transposition (also using AREP) of the expression above as

$$\begin{aligned}\mathrm{DCT\text{-}II}_8 &= [(2,5)(4,7)(6,8),8] \cdot (\mathrm{diag}(1,\tfrac{1}{\sqrt{2}}) \oplus \mathrm{R}_{\frac{3}{8}\pi} \oplus \mathrm{R}_{\frac{15}{16}\pi} \oplus \mathrm{R}_{\frac{21}{16}\pi}) \\ &\cdot[(2,4,7,3,8),8] \cdot ((\mathrm{DFT}_2 \otimes \mathbf{1}_3) \oplus \mathbf{1}_2) \\ &\cdot[(5,6),8] \cdot (\mathbf{1}_4 \oplus \tfrac{1}{\sqrt{2}} \cdot \mathrm{DFT}_2 \oplus \mathbf{1}_2) \cdot [(2,3,4,5,8,6,7),8] \\ &\cdot(\mathbf{1}_2 \otimes ((\mathrm{DFT}_2 \oplus \mathbf{1}_2) \cdot [(2,3),4] \cdot (\mathbf{1}_2 \otimes \mathrm{DFT}_2))) \cdot [(1,8,6,2)(3,4,5,7),8].\end{aligned}$$

The general form of the symmetries for the DCTs and their algebraic derivation can be found in Püschel and Moura (2001).

6.3. Discrete Cosine Transform, type IV

The (unscaled) discrete cosine transform of type IV is defined by the matrix

$$\text{DCT-IV}_n = [\cos((2k+1)(2\ell+1)\pi/4n) \mid 0 \le k, \ell < n].$$

For $n = 8$ we find a mon-irred symmetry (ϕ, ψ) with dihedral symmetry group $\mathsf{D}_{32} = \langle x, y \mid x^{16} = y^2 = 1, x^y = x^{-1}\rangle$,

$$\begin{aligned} \phi : x &\mapsto [(1,3,5,7,8,6,4,2),(1,1,1,1,1,1,1,-1)],\\ y &\mapsto [(2,3)(4,5)(6,7),(1,1,1,1,1,1,1,-1)],\\ \psi : x &\mapsto M_1,\ y \mapsto M_2. \end{aligned}$$

Using the shorthand notation $c_k = \cos(k\pi/16)$ and $s_k = \sin(k\pi/16)$, the matrices M_1 and M_2 are given by

$$M_1 = \begin{bmatrix} c_2 & 0 & 0 & 0 & 0 & 0 & 0 & s_2 \\ 0 & c_6 & 0 & 0 & 0 & 0 & s_6 & 0 \\ 0 & 0 & c_{10} & 0 & 0 & s_{10} & 0 & 0 \\ 0 & 0 & 0 & c_{14} & s_{14} & 0 & 0 & 0 \\ 0 & 0 & 0 & s_{18} & c_{18} & 0 & 0 & 0 \\ 0 & 0 & s_{22} & 0 & 0 & c_{22} & 0 & 0 \\ 0 & s_{26} & 0 & 0 & 0 & 0 & c_{26} & 0 \\ s_{30} & 0 & 0 & 0 & 0 & 0 & 0 & c_{30} \end{bmatrix}, \quad M_2 = \begin{bmatrix} c_1 & 0 & 0 & 0 & 0 & 0 & 0 & s_1 \\ 0 & c_3 & 0 & 0 & 0 & 0 & s_3 & 0 \\ 0 & 0 & c_5 & 0 & 0 & s_5 & 0 & 0 \\ 0 & 0 & 0 & c_7 & s_7 & 0 & 0 & 0 \\ 0 & 0 & 0 & s_9 & c_9 & 0 & 0 & 0 \\ 0 & 0 & s_{11} & 0 & 0 & c_{11} & 0 & 0 \\ 0 & s_{13} & 0 & 0 & 0 & 0 & c_{13} & 0 \\ s_{15} & 0 & 0 & 0 & 0 & 0 & 0 & c_{15} \end{bmatrix}.$$

The representation ψ is a permuted direct sum of irreducibles of degree 2: $\text{cbs}(M_1) = \text{cbs}(M_2) = (1\,8|2\,7|3\,6|4\,5)$.

Based on this symmetry we find the factorization

$$\begin{aligned} \text{DCT-IV}_8 = {} & [(1,2,8)(3,6,5),(1,-1,1,1,1,-1,1,1)] \\ & \cdot(\mathbf{1}_2 \otimes ((\mathbf{1}_2 \oplus \tfrac{1}{\sqrt{2}} \cdot \text{DFT}_2) \cdot [(3,4),4] \cdot (\text{DFT}_2 \otimes \mathbf{1}_2))) \\ & \cdot[(1,3)(2,4)(5,7)(6,8),8] \cdot (\mathbf{1}_4 \oplus \mathrm{R}_{\frac{15}{8}\pi} \oplus \mathrm{R}_{\frac{11}{8}\pi}) \cdot (\text{DFT}_2 \otimes \mathbf{1}_4) \\ & \cdot[(3,5,7)(4,6,8),8] \cdot (\mathrm{R}_{\frac{31}{32}\pi} \oplus \mathrm{R}_{\frac{19}{32}\pi} \oplus \mathrm{R}_{\frac{27}{32}\pi} \oplus \mathrm{R}_{\frac{23}{32}\pi}) \\ & \cdot[(1,8,5,6,3,2)(4,7),8]. \end{aligned}$$

The factorization was generated in 6.8 seconds.

6.4. Hartley Transform

The (unscaled) discrete Hartley transform DHT_n is defined by the matrix

$$\text{DHT}_n = [\cos(2k\ell\pi/n) + \sin(2k\ell\pi/n) \mid 0 \le k, \ell < n].$$

We find a perm-irred symmetry (ϕ, ψ) with dihedral symmetry group $\mathsf{D}_{16} = \langle x, y \mid x^8 = y^2 = 1, x^y = x^{-1}\rangle$. We give only ϕ:

$$\phi :\ x \mapsto [(1,2,3,4,5,6,7,8),8], \quad y \mapsto [(2,8)(3,7)(4,6),8].$$

The corresponding factorization is given by

$$\begin{aligned}
\mathrm{DHT}_8 \;=\; & [(1,8)(2,4)(3,6)(5,7),8] \\
& \cdot(\mathbf{1}_2 \otimes ((\mathbf{1}_2 \otimes \mathrm{DFT}_2) \cdot [(2,3),4] \cdot (\mathrm{DFT}_2 \oplus \mathbf{1}_2))) \cdot [(2,7,6,8,5,4,3),8] \\
& \cdot(\mathbf{1}_4 \oplus -\tfrac{1}{\sqrt{2}} \cdot \mathrm{DFT}_2 \oplus \mathbf{1}_2) \cdot [(5,6),8] \cdot ((\mathrm{DFT}_2 \otimes \mathbf{1}_3) \oplus \mathbf{1}_2) \\
& \cdot[(2,5,3,6,4)(7,8),(1,-1,-\sqrt{2},-\sqrt{2},\sqrt{2},\sqrt{2},-1,-1)] \\
& \cdot(\mathbf{1}_6 \oplus \mathrm{DFT}_2) \cdot [(2,5,8,7,3,4),8].
\end{aligned}$$

The factorization was generated in 1.1 seconds.

The DHT_8 also has another perm-irred symmetry (ϕ, ψ) with symmetry group D_{16}. We give again only ϕ:

$$\phi:\ x \mapsto [(1,2,3,8,5,6,7,4),8], \quad y \mapsto [(2,4)(3,7)(6,8),8].$$

The resulting factorization is very similar to the one above.

Furthermore, DHT_8 has a mon-mon symmetry with a symmetry group of size 256 and the structure $\mathsf{Z}_2 \ltimes (\mathsf{Z}_2 \times \mathsf{Z}_2 \times \mathsf{Z}_2 \times \mathsf{Z}_2 \times \mathsf{D}_8)$, where $H \ltimes N$ denotes the semidirect product with normal subgroup N. The resulting factorization is quite different from the one above:

$$\begin{aligned}
\mathrm{DHT}_8 \;=\; & [(1,4,6,7)(2,8,5,3),(1,-1,-1,1,1,1,-1,-1)] \cdot (\mathbf{1}_4 \otimes \mathrm{DFT}_2) \\
& \cdot[(1,5,7,8,2,3,6),(\sqrt{2},-\sqrt{2},1,1,1,1,1,1)] \\
& \cdot(\mathbf{1}_2 \oplus \mathrm{DFT}_2 \oplus -(\mathbf{1}_2 \otimes \mathrm{DFT}_2)) \\
& \cdot[(1,8,2,6,5,7,4),8] \cdot (\mathbf{1}_4 \otimes \mathrm{DFT}_2) \\
& \cdot[(1,5,6,2)(3,7,4),(1,1,-1,-1,1,1,-1,-1)].
\end{aligned}$$

Generating this factorization took 2.4 seconds.

We also chose the Hartley transform to illustrate the runtime behavior of the three steps in the factorization algorithm (see Section 2) as the transform size increases. Table 3 displays the runtime results (in seconds) for a decomposition via perm-irred symmetry (left table) and via mon-mon symmetry (right table). The size of the group found is in the second row and bold-faced, the runtimes for the three steps in the factorization algorithm are given in rows 3–5: find symmetry, decompose symmetry, and combine decompositions. The bottom line shows the total runtime needed to generate the factorization. We note that in all cases the DHT was fully decomposed, i.e., the resulting structural expression did not contain any subblocks of size larger than 2×2.

For the decomposition via perm-irred symmetry we observe a steep increase in runtime for finding the symmetry, whereas decomposing the symmetry is rather fast due to the modest group sizes (a dihedral group in all cases). In contrast, the decomposition via mon-mon symmetry finds the symmetry very fast and suffers from the runtimes for decomposing the symmetry, which is due to the large group size. According to our experience, Table 3 provides examples that serve as good representatives for the performance of our algorithms.

6.5. Haar Transform

The Haar transform HT_{2^k} is defined recursively by:

$$\mathrm{HT}_2 = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}, \quad \mathrm{HT}_{2^{k+1}} = \begin{bmatrix} \mathrm{HT}_{2^k} & \otimes [1 \;\; 1] \\ 2^{k/2} \cdot \mathbf{1}_{2^k} & \otimes [1 \;\; -1] \end{bmatrix}, \quad k \geq 1.$$

Table 3. Runtime profile for decomposing a DHT across different sizes via perm-irred symmetry (left) and mon-mon symmetry (right); the runtimes are given in seconds.

DHT size	8	16	32
group size	**16**	**32**	**64**
symmetry	0.8	64	4872
decompose	0.1	0.8	2.1
combine	0.2	0.5	2.2
total time	1.1	65	4876

DHT size	8	16	32	64	128	256
group size	**256**	**256**	**512**	**1024**	**2048**	**4096**
symmetry	0	0	0.6	4.8	46	4028
decompose	2.4	5.9	15	1319	6469	34262
combine	0	0	0.5	2.7	16	138
total time	2.4	6.0	16	1381	6531	38428

A fast algorithm for the Haar transform follows directly from the definition. For $k = 3$ we build the corresponding matrix HT_8. The transpose of HT_8 has a perm-irred symmetry (i.e., HT_8 has an irred-perm symmetry). The symmetry group is the iterated wreath product $(\mathsf{Z}_2 \wr \mathsf{Z}_2) \wr \mathsf{Z}_2$ of size 128 (Foote *et al.*, 2000). By transposing the resulting factorization we obtain the following factorization of HT_8.

$$\begin{aligned}\mathrm{HT}_8 \quad = \quad & [(1,8,6,4,2,7,5,3),8] \\ & \cdot(\mathrm{diag}(-\sqrt{2},\sqrt{2}) \oplus \mathbf{1}_4 \oplus \mathrm{DFT}_2) \cdot [(1,5,4,8,6,3,7,2),8] \\ & \cdot(\mathbf{1}_2 \otimes ([(1,2),4] \cdot (\mathrm{DFT}_2 \oplus 2 \cdot \mathbf{1}_2) \cdot [(2,3),4] \cdot (\mathbf{1}_2 \otimes \mathrm{DFT}_2))) \\ & \cdot[(1,8,4,7)(3,6,2,5),(1,1,1,1,-1,-1,-1,-1)].\end{aligned}$$

Generating this factorization took 6.5 seconds.

7. Acknowledgments

A major part of the presented results has been developed when the authors where with Prof. Beth at the Institute of Algorithms and Cognitive Systems at the University of Karlsruhe, Germany. The authors are indebted to Prof. Beth for his support that made this work possible. The authors also want to thank Prof. Johnson (Drexel University) for helpful discussions and the anonymous reviewers for many helpful comments.

References

Apple, G., Wintz, P. (1970). Calculation of Fourier Transforms on Finite Abelian Groups. *IEEE Trans. on Information Theory*, **IT-16**(2):233–34.

Auslander, L., Feig, E., Winograd, S. (1984). Abelian Semi-simple Algebras and Algorithms for the Discrete Fourier Transform. *Advances in Applied Mathematics*, **5**:31–55.

Beth, T. (1984). *Verfahren der Schnellen Fouriertransformation*. Teubner.

Clausen, M., Baum, U. (1993). *Fast Fourier Transforms*. BI-Wiss.-Verl.

Cooley, J. W., Tukey, J. W. (1965). An Algorithm for the Machine Calculation of Complex Fourier Series. *Math. of Computation*, **19**:297–301.

Curtis, W. C., Reiner, I. (1962). *Representation Theory of Finite Groups*. Interscience.

Egner, S. (1997). *Zur Algorithmischen Zerlegungstheorie Linearer Transformationen mit Symmetrie*. PhD thesis, Universität Karlsruhe, Informatik.

Egner, S., Johnson, J., Padua, D., Püschel, M., Xiong, J. (2001). Automatic Derivation and Implementation of Signal Processing Algorithms. *ACM SIGSAM Bulletin Communications in Computer Algebra*, **35**(2):1–19.

Egner, S., Püschel, M. (1998). *AREP – Constructive Representation Theory and Fast Signal Transforms*. GAP share package. `http://www.ece.cmu.edu/~smart/arep/arep.html`.

Egner, S., Püschel, M. (2001). Automatic Generation of Fast Discrete Signal Transforms. *IEEE Trans. on Signal Processing*, **49**(9):1992–2002.

Elliott, D. F., Rao, K. R. (1982). *Fast Transforms: Algorithms, Analyses, Applications.* Academic Press.

Foote, R., Mirchandi, G., Rockmore, D., Healy, D., Olson, T. (2000). A Wreath Product Approach to Signal and Image Processing: Part I — Multiresolution Analysis. *IEEE Trans. on Signal Processing*, **48**(1):102–132.

GAP (1997). *GAP – Groups, Algorithms, and Programming.* The GAP Team, University of St. Andrews, Scotland. `http://www-gap.dcs.st-and.ac.uk/~gap/`.

Garey, M. R., Johnson, D. (1979). *Computers and Intractability.* W. H. Freeman and Co.

Huppert, B. (1983). *Endliche Gruppen*, volume I. Springer.

James, G. D., Kerber, A. (1981). *The Representation Theory of the Symmetric Group*, volume 16 of *Encyclopedia of Mathematics.* Addison-Wesley.

Karpovsky, G. D., Trachtenberg, E. A. (1977). Fast fourier transforms on finite non-abelian groups. *IEEE Trans. Comput.*, **C-26**(10):1028–1030.

Leon, J. (1991). Permutation Group Algorithms Based on Partitions, I: Theory and Algorithms. *Journal of Symbolic Computation*, **12**(4/5):533–583.

Maslen, D., Rockmore, D. (1995). Generalized FFTs – A survey of some recent results. In *Proceedings of IMACS Workshop in Groups and Computation*, volume 28, pages 182–238.

Maslen, D., Rockmore, D. (1997). Separation of variables for the efficient computation of Fourier transforms on finite groups, 1. *Journal of the American Mathematical Society*, **10**(1):169–214.

Minkwitz, T. (1993). *Algorithmensynthese für lineare Systeme mit Symmetrie.* PhD thesis, Universität Karlsruhe, Informatik.

Minkwitz, T. (1995). Algorithms Explained by Symmetry. *Lecture Notes on Computer Science*, **900**:157–167.

Moura, J. M. F., Johnson, J., Johnson, R. W., Padua, D., Prasanna, V., Püschel, M., Veloso, M. M. (1998). SPIRAL: A Generator for Platform-Adapted Libraries of Signal Processing Algorithms. `http://www.ece.cmu.edu/~spiral/`.

Parker, R. (1984). The Computer Calculation of Modular Characters (the MeatAxe). In *M. D. Atkinson (ed.), Computational Group Theory, Proceedings LMS Symposium on Computational Group Theory, Durham 1982, Academic Press.*

Püschel, M. (1998). *Konstruktive Darstellungstheorie und Algorithmengenerierung.* PhD thesis, Universität Karlsruhe, Informatik. Also available in English as Tech. Rep. Drexel-MCS-1999-1, Drexel University, Philadelphia.

Püschel, M. (2002). Decomposing Monomial Representations of Solvable Groups. *Journal of Symbolic Computation.* To appear.

Püschel, M., Moura, J. M. F. (2001). The Algebraic Approach to the Discrete Cosine and Sine Transforms and their Fast Algorithms. Submitted for publication.

Rao, K. R., Yip, P. (1990). *Discrete Cosine Transform: Algorithms, Advantages, Applications.* Academic Press.

Wedderburn, J. H. M. (1907). On Hypercomplex Numbers. *Proceedings of the London Mathematical Society.*