Automatic Generation of Fast Discrete Signal Transforms

Sebastian Egner and Markus Püschel

***Abstract*—This paper presents an algorithm that derives fast versions for a broad class of discrete signal transforms symbolically. The class includes but is not limited to the discrete Fourier and the discrete trigonometric transforms. This is achieved by finding fast sparse matrix factorizations for the matrix representations of these transforms. Unlike previous methods, the algorithm is entirely automatic and uses the defining matrix as its sole input. The sparse matrix factorization algorithm consists of two steps: First, the "symmetry" of the matrix is computed in the form of a pair of group representations; second, the representations are stepwise decomposed, giving rise to a sparse factorization of the original transform matrix. We have successfully demonstrated the method by computing automatically efficient transforms in several important cases: For the DFT, we obtain the Cooley–Tukey FFT; for a class of transforms including the DCT, type II, the number of arithmetic operations for our fast transforms is the same as for the best-known algorithms. Our approach provides new insights and interpretations for the structure of these signal transforms and the question of why fast algorithms exist. The sparse matrix factorization algorithm is implemented within the software package AREP.**

***Index Terms*—Discrete cosine transform, discrete Fourier transform, fast algorithm, group representations, monomial representations, symmetry, trigonometric transforms.**

I. Introduction

FAST algorithms for discrete signal transforms have been a major research topic in the last decades leading to a large number of publications. Because of their wide-spread applications in digital signal processing, particular effort has been spent on the discrete Fourier transform (DFT) and the different types of trigonometric transforms, i.e., discrete cosine and sine transforms (DCTs and DSTs), as classified by Wang and Hunt [1]. Important algorithms for the DFT include the "fast Fourier transform" (FFT) found by Cooley and Tukey (first discovered by Gauss [2]) [3], Rader's algorithm for prime size [4], Winograd's algorithms [5], as well as [6]–[8]. An overview on FFT algorithms can be found in [9] or [10]. Important algorithms for the trigonometric transforms were found by Chen *et al.* [11], Wang [12], Yip and Rao [13], [14], Vetterli and Nussbaumer [8], Lee [15], Feig [16], Chan and Ho [17], Steidl and Tasche [18], and Feig and Winograd [19].

Most of the algorithms cited above are given as a factorization of the respective transform matrix into a product of highly structured, sparse matrices. If an algorithm is given another way, e.g., by equations, it is possible to rewrite the algorithm in the form of a sparse matrix product.

All of these algorithms have been found by insightful manipulation of the entries of the transform matrices using algebraic relationships of these numbers. In some papers, these relationships have been referred to as "symmetry." Several questions remain unanswered. Is there a general mathematical principle behind these algorithms, i.e., matrix factorizations? What is the appropriate definition of symmetry that accounts for the existence of the algorithms? Is it possible to automate the process of finding algorithms? For the DFT, the first two questions have been answered, as we will briefly discuss in the next subsection, since it marks the starting point for our results.

In this paper, we present the mathematical background and the algorithm to automatically generate fast algorithms, given as sparse matrix factorizations, for a large class of discrete signal transforms using techniques from group representation theory. In particular, we present the following.

- *An appropriate definition of "symmetry" that catches redundancy contained in the transform matrix and connects it to group representations.* Furthermore, the symmetry has an intuitive interpretation in terms of signal processing. As we will see, this definition of symmetry generalizes the well-known property of the DFT diagonalizing the cyclic shift.
- *An algorithm that 1) finds the symmetry of a matrix and 2) uses it to derive a sparse matrix factorization.* The algorithm has been implemented and can be used as a discover tool for fast transforms.
- *The successful application of the factorization algorithm to a large class of transforms.* In many cases, the generated fast transforms are similar to, or have the same arithmetic cost (operations count), as the best-known algorithms.

Taken together, we provide a unifying framework that shows that a large class of the best known fast algorithms for different transforms are all special instances of the same common principle. Thus, we shed new light on fast algorithms, put them into a common context, and give insight into their algebraic structure.

A. Signal Transforms and Group Representations

The use of finite groups and their representations is not new in signal processing. The most important example is the DFT

Manuscript received June 1, 2000; revised May 30, 2001. M. Püschel was supported by the National Science Foundation through Award 9988296 and by the Defense Advanced Research Projects Agency through research Grant DABT63-98-1-0004, administered by the Army Directorate of Contracting. The associate editor coordinating the review of this paper and approving it for publication was Dr. Xiang-Gen Xia.

S. Egner is with the Philips Research Laboratories, Eindhoven, The Netherlands (e-mail: sebastian.egner@philips.com).

M. Püschel is with the Department of Electrical and Computer Engineering, Carnegie Mellon University, Pittsburgh, PA 15213 USA (e-mail: pueschel@ece.cmu.edu).

Publisher Item Identifier S 1053-587X(01)07061-1.

and its connection to cyclic groups and their regular representations. This connection has been used to derive and explain the structure of the Cooley–Tukey FFT [20], [21]. Generalization to arbitrary groups, also known as Fourier analysis on groups, has lead to a rich class of transforms, which, however, have found no significant applications in signal processing [22]–[25]. An exception might be the recent paper [26], where nonregular representations of so-called wreath-product groups have been proposed for multi-resolution image processing.

The crucial step to capture in the group representation framework a broader class of signal transforms, including the cosine and sine transforms, is to leave the domain of "regular" representations in favor of the larger class of "monomial" representations. The idea has its roots in the work of Minkwitz [27], [28], and has been further developed by the authors in [29]–[33], which forms the basis for this paper. We provide the tools to investigate a *given* transform for group representation properties and, when appropriate, factorize the transform, thus obtaining a fast algorithm.

B. Approach

The approach for generating a fast algorithm for a given signal transform, which is given as a matrix M, consists basically of two steps. In the first step, the "symmetry" of M is computed. The "symmetry" is a pair of group representations representing an invariance property of M (cf. Section III). In the second step, the group representations are decomposed stepwise. This gives rise to factorized decomposition matrices and determines a factorization of M as a product of sparse matrices (cf. Section IV). The factorization represents a fast algorithm for the transform M. Intuitively speaking, the "symmetry" captures a large part of the redundancy contained in M, and the decomposition of the representations turns the redundancy into a fast algorithm.

C. Organization of the Paper

In Section II, we introduce our notation for representing structured matrices and present the basic terms of group representations that are necessary to understand our approach for obtaining matrix factorizations. We emphasize the concepts and the methodology rather than explaining the technical details. The notion of "symmetry" of a matrix is defined in Section III, and Section IV explains how a symmetry can be used to derive a matrix factorization. In Section V, we apply the matrix factorization algorithm to the Fourier transform, cosine, and sine transforms of different types, the Hartley transform, and the Haar transform. We compare the structure and arithmetic cost of the algorithms that we derive to the structure and cost of well-known algorithms from the literature. We conclude the paper with a brief summary and an outlook for future research in Section VI.

II. Mathematical Background

In this section, we present the basic notation of matrices and group representations we are going to use. For further information on representation theory, see introductory books such as [34].

A. Matrices

We use the following notation to represent matrices. $[\sigma, n]$ is the $(n \times n)$-permutation matrix corresponding to the permutation σ, which is given in cycle notation, e.g., $\sigma = (1, 2, 3)(4, 6)$ means $\sigma(1) = 2$, $\sigma(2) = 3$, $\sigma(3) = 1$, $\sigma(4) = 6$, $\sigma(5) = 5$, $\sigma(6) = 4$ with corresponding (6×6)-matrix

$$[(1, 2, 3)(4, 6), 6] = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix}.$$

Note that it is necessary to supply the size n of a permutation matrix in $[\sigma, n]$ since fixed points are omitted in cycle notation (i.e., there is a difference between $[(1, 2), 2]$ and $[(1, 2), 3]$). We prefer cycle notation because one can read off the order and the fixed points of a permutation immediately. With $\mathbf{1}_n$, we denote the identity matrix of size n, $\mathrm{diag}(L)$ is a diagonal matrix with the list L on the diagonal. A *monomial* matrix (sometimes called scaled permutation matrix) has exactly one nonzero entry in every row and column and is represented as $[\sigma, L] = [\sigma, \mathrm{length}(L)] \cdot \mathrm{diag}(L)$, e.g.,

$$[(1, 2, 3), (-1, 1, 2)] = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 2 \\ -1 & 0 & 0 \end{bmatrix}$$

i.e., the list L scales the columns of the matrix. The operator $\otimes$ denotes the Kronecker (or tensor) product of matrices, and $\oplus$ denotes the direct sum

$$A \oplus B = \begin{bmatrix} A & \mathbf{0} \\ \mathbf{0} & B \end{bmatrix}$$

where $\mathbf{0}$ is an all-zero matrix of appropriate size.

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

is the rotation matrix for angle α, and

$$\mathrm{DFT}_n = [\omega_n^{k\ell} | k, \ell = 0 \ldots n-1]$$

where $\omega_n = e^{2\pi j/n}$, denotes the discrete Fourier transform of size n.

B. Groups and Representations

In this paper, essentially only two types of groups will appear: the cyclic group of size n, written as $\mathsf{Z}_n = \{1, x, \ldots, x^{n-1}\}$ or, by generators and relations, as $\mathsf{Z}_n = \langle x | x^n = 1 \rangle$ and the dihedral group of size $2n$ denoted by $\mathsf{D}_{2n} = \{1, x, \ldots, x^{n-1}, y, yx, \ldots, yx^{n-1}\} = \langle x, y | x^n = y^2 = 1, y^{-1}xy = x^{-1} \rangle$.

A *representation* of a group G (over $\mathbb{C}$) is a homomorphism

$$\phi: G \to \mathrm{GL}_n(\mathbb{C})$$

of G into the group $\mathrm{GL}_n(\mathbb{C})$ of invertible $(n \times n)$-matrices over the complex numbers $\mathbb{C}$. n is called the *degree* of ϕ. Dealing with representations is nothing but dealing with groups of matrices. If ϕ is a representation of G, then $\phi(G)$ is a matrix group,

and, vice–versa, every matrix group can be viewed as a representation of itself. If $A \in \mathrm{GL}_n(\mathbb{C})$, then ϕ^A: $g \mapsto A^{-1} \cdot \phi(g) \cdot A$ is called the *conjugate* of ϕ by A. The representations ϕ and ϕ^A are called *equivalent*. If ϕ and ψ are representations of G, then the representation $\phi \oplus \psi$: $g \mapsto \phi(g) \oplus \psi(g)$ is called the *direct sum* of ϕ and ψ. The direct sum of n representations $\phi_1, \ldots, \phi_n$ is defined analogously. The representation ϕ is *irreducible* if it cannot be conjugated to be a direct sum.

Theorem 1 (Maschke): Every representation ϕ (over $\mathbb{C}$) of a group G can be decomposed into a direct sum of irreducible representations by conjugation with a suitable matrix A

$$\phi^A = \rho_1 \oplus \cdots \oplus \rho_k.$$

The ρ_i are uniquely determined up to equivalence and up to a permutation of $\rho_1, \ldots, \rho_k$.

In other words, Theorem 1 tells us how far a finite group of matrices can be simultaneously block diagonalized. The matrix A in Theorem 1 is not uniquely determined and is called a *decomposition matrix* for ϕ.

ϕ is called a *permutation* representation if all images $\phi(g)$ are permutation matrices, and ϕ is called a *monomial* representation if all images $\phi(g)$ are monomial matrices. Every permutation representation is also a monomial representation.

The following example states the interpretation of the DFT in terms of representation theory.

Example 1: It is a known fact that DFT_n maps the cyclic shift (and all its powers) in the time-domain into a phase change in the frequency-domain. In our notation

$$[(1, 2, \ldots, n), n] \cdot \mathrm{DFT}_n = \mathrm{DFT}_n \cdot \mathrm{diag}(1, \omega_n, \ldots, \omega_n^{n-1}).$$

In terms of representation theory, DFT_n decomposes the permutation representation ϕ: $x \mapsto [(1, 2, \ldots, n), n]$ of the cyclic group $G = \mathrm{Z}_n = \langle x | x^n = 1 \rangle$ into the direct sum $\phi^{\mathrm{DFT}_n} = \rho_1 \oplus \cdots \oplus \rho_n$, where the irreducible representations are $\rho_k = x \mapsto \omega_n^{k-1}$.

III. Symmetry of a Matrix

The notion of symmetry has a two-fold purpose. First, it catches the redundancy contained in the matrix M; second, it establishes the connection to representation theory, which enables the application of algebraic methods to factorize M, as sketched in Section IV.

We consider an arbitrary rectangular matrix $M \in \mathbb{C}^{m \times n}$. A *symmetry* of M is a pair (ϕ_1, ϕ_2) of representations of the same group G satisfying

$$\phi_1(g) \cdot M = M \cdot \phi_2(g), \quad \text{for all} \quad g \in G.$$

We call G a *symmetry group* of M. We will use a shorthand notation and write $\phi_1 \xrightarrow{M} \phi_2$.

A symmetry (ϕ_1, ϕ_2) has a very natural interpretation in terms of signal processing if M is a discrete signal transform that we multiply from the left.

For all $g \in G$, a multiplication with $\phi_2(g)$ in the time-domain corresponds to a multiplication with $\phi_1(g)$ in the frequency domain.

With the general definition above, however, every matrix has arbitrary many symmetries. If, for example, M is an invertible $(n \times n)$-matrix and ϕ is any representation of degree n of a group G, then M has the symmetry (ϕ, ϕ^M). Thus, in order to catch the redundancy contained in M, we will consider several "types" of symmetry arising from restrictions on the representations ϕ_1, ϕ_2.

1) *Mon-Irred Symmetry*: ϕ_1 is monomial, and ϕ_2 is a direct sum of irreducible representations. If ϕ_1 is even a permutation representation, then we will also speak of *perm-irred symmetry*.
2) *Mon-Mon Symmetry*: ϕ_1 and ϕ_2 are monomial. If ϕ_1 and ϕ_2 are both even permutation representations, then we will also speak of *perm-perm symmetry*.

In words, the matrix M has a mon-mon symmetry if there are nontrivial monomial matrices L, R such that $L \cdot M = M \cdot R$. Correspondingly, the matrix M has a mon-irred symmetry if M is a decomposition matrix for a monomial representation ϕ. The rationale for considering the types of symmetry above will become clear in Section IV. Of course, one could also consider an irred-mon symmetry where ϕ_2 is monomial and ϕ_1 is decomposed. Since transposition of a matrix with irred-mon symmetry yields a matrix with mon-irred symmetry, we will restrict to the latter symmetry type. Finding symmetry of the types above is a difficult combinatorial problem and a main topic of [29] and [32]. In fact, even computing the perm-perm symmetry has a complexity that is not lower than testing graphs isomorphism, which is known to be hard [35]. However, for matrices originating from signal transformations, it is often practical to compute the symmetry because they contain many different entries, which reduces the search space.

Example 2: Example 1 shows that the DFT_n has the symmetry group $G = \mathrm{Z}_n = \langle x | x^n = 1 \rangle$ with symmetry (ϕ_1, ϕ_2):

$$\phi_1\text{: } x \mapsto [(1, 2, \ldots, n), n]$$
$$\phi_2\text{: } x \mapsto \mathrm{diag}(1, \omega_n, \ldots, \omega_n^{n-1}).$$

Note that (ϕ_1, ϕ_2) is a mon-irred symmetry (even a perm-irred symmetry) as well as a mon-mon symmetry.

IV. Matrix Factorization

Now, we explain how to factorize a given matrix M, which has an arbitrary symmetry (ϕ_1, ϕ_2). First, the representations ϕ_1, ϕ_2 are decomposed with matrices A_1, A_2, respectively. This gives rise to two decomposed representations $\rho_1 = \phi_1^{A_1}$, $\rho_2 = \phi_2^{A_2}$. Second, the matrix $D = A_1^{-1} \cdot M \cdot A_2$ is computed to obtain the commutative diagram in Fig. 1.

Altogether, we obtain the factorization

$$M = A_1 \cdot D \cdot A_2^{-1}. \tag{1}$$

From representation theory, we know that D is a sparse matrix (cf. [31, Th. 1.48, iv]), but the question of sparsity remains regarding the matrices A_1 and A_2. The factorization in (1) is useful only if the decomposition matrices A_1 and A_2 can themselves be determined as a product of sparse matrices. This is possible for monomial representations (with certain restrictions on the symmetry group G), as has been developed in the thesis

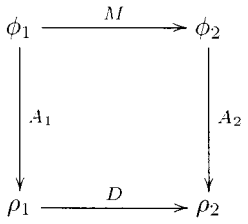

Fig. 1. Factorization of the matrix M with symmetry (ϕ_1, ϕ_2).

research [31], [33], and justifies the consideration of the two types of symmetry described in Section III.

1) Mon-mon symmetry: A_1 and A_2 are decomposition matrices of monomial representations.
2) Mon-irred symmetry: A_1 is a decomposition matrix of a monomial representation, and A_2 is the identity since ϕ_2 is already decomposed.

In fact, we will slightly relax the definition of mon-irred symmetry and allow A_2 to be any permutation matrix, which means that ϕ_2 is a *permuted* direct sum of irreducible representations. The factorization of a decomposition matrix for a monomial representation ϕ arises from an algorithm that stepwise decomposes ϕ along a chain of normal subgroups using recursion formulas for the decomposition matrices [33]. The recursion formula essentially determines the structure of the matrix factorizations that we will present in Section V.

The algorithm for factorizing a matrix with symmetry follows Fig. 1 and reads as follows.

Algorithm 1: Given a matrix M to be factorized into a product of sparse matrices.

1) Determine a suitable symmetry (ϕ_1, ϕ_2) of M.
2) Decompose ϕ_1 and ϕ_2 stepwise, and obtain (factorized) decomposition matrices A_1, A_2.
3) Compute the sparse matrix $D = A_1^{-1} \cdot M \cdot A_2$.

Result: $M = A_1 \cdot D \cdot A_2^{-1}$ is a factorization of M into a product of sparse matrices. This is a fast algorithm for evaluating the linear transformation $x \mapsto M \cdot x$.

Algorithm 1 is implemented in the library AREP [36], which is a GAP share package for symbolic computation with group representations and structured matrices. GAP [37] is a computer algebra system for symbolic computation with groups. AREP has been created as part of the thesis research [29], [30].

In the Appendix, we provide an overview of Steps 1 and 2. A comprehensive treatment including the mathematical background and all technical details can be found in [29] and [31]–[33]. In the following, we will concentrate on how the combinatorial search in Step 1 and the algebraic decomposition of Step 2 can be combined to automatically generate fast signal transforms. In particular, we are interested in answering the following questions.

- To which transforms is our approach applicable?
- What are the symmetry properties found?
- How do our generated algorithms compare with algorithms known from literature?

First, we start with two brief initial examples applying Algorithm 1 to the DFT and a circulant matrix. A more detailed version of Example 4 can be found in the Appendix.

Example 3: Let $M = \mathrm{DFT}_4$. M has the mon-irred resp. perm-irred symmetry

$$\phi_1 \colon x \mapsto [(1, 2, 3, 4), 4]$$
$$\phi_2 \colon x \mapsto \mathrm{diag}(1, \omega_4, \omega_4^2, \omega_4^3)$$

(cf. Example 2). ϕ_2 is already decomposed, and hence, $A_2 = \mathbf{1}_4$. Decomposing ϕ_1 stepwise yields the decomposition matrix

$$A_1 = (\mathrm{DFT}_2 \otimes \mathbf{1}_2) \cdot \mathrm{diag}(1, 1, 1, \omega_4) \cdot (\mathbf{1}_2 \otimes \mathrm{DFT}_2) \cdot [(2, 3), 4].$$

We compute $D = A_1^{-1} \cdot M \cdot A_2 = \mathbf{1}_4$ and get the Cooley–Tukey factorization $M = A_1$.

Example 4: Consider the circulant (4×4)-matrix

$$M = \begin{bmatrix} 1 & 2 & 3 & 4 \\ 4 & 1 & 2 & 3 \\ 3 & 4 & 1 & 2 \\ 2 & 3 & 4 & 1 \end{bmatrix}.$$

M has the mon-mon (even perm-perm) symmetry

$$\phi_1 \colon x \mapsto [(1, 2, 3, 4), 4]$$
$$\phi_2 \colon x \mapsto [(1, 2, 3, 4), 4].$$

Decomposing ϕ_1 and ϕ_2 into a direct sum of irreducible representations yields the decomposition matrices

$$A_1 = A_2 = \mathrm{DFT}_4 = (\mathrm{DFT}_2 \otimes \mathbf{1}_2) \cdot \mathrm{diag}(1, 1, 1, \omega_4) \cdot (\mathbf{1}_2 \otimes \mathrm{DFT}_2) \cdot [(2, 3), 4].$$

We compute $D = A_1^{-1} \cdot M \cdot A_2 = \mathrm{diag}(a, b, c, d)$ with complex numbers a, b, c, d (whose actual values are not important here) and obtain the well-known factorization of the cyclic convolution

$$M = \mathrm{DFT}_4 \cdot D \cdot \mathrm{DFT}_4^{-1}.$$

(We detail Steps 1 and 2 in the Appendix.)

V. Examples

In this section, we apply Algorithm 1 to a number of signal transforms. The following factorizations have been generated from the respective transform *entirely automatically* using the GAP share package AREP [36], [37], which contains an implementation of Algorithm 1. Even the LaTeX expressions displayed below have been generated verbatim as they are.

We show the symmetry of the considered transforms, state the number of arithmetic operation needed by our derived fast algorithms, and compare them with algorithms known from literature. We want to emphasize that the symmetries themselves are of interest since the fast algorithms that we derive owe their existence to the symmetry in the same way as the Cooley–Tukey FFT owes its existence to the fact that the DFT diagonalizes the cyclic shift. The algebraic structure of the fast algorithms found is due to the recursion formula for decomposing monomial representations (done in Step 2 of Algorithm 1), which is subject of Theorem 2 in the Appendix.

First, we want to say some words about how AREP deals with structured matrices. AREP does *symbolic* computation with ma-

trices, which means it stores and manipulates expressions representing matrices rather than the matrices themselves. An expression is something like $(\mathrm{DFT}_4 \otimes \mathbf{1}_{24}) \oplus \mathbf{1}_{17}$, which can be stored and manipulated more efficiently than the large matrix it represents. Of course, an expression can always be converted into a real matrix if desired. While building the structured matrices shown below (in particular in Step 2 of Algorithm 1), AREP simplifies according to specified rules. For example, monomial or permutation matrices are extracted from Kronecker products or direct sums and multiplied if they are adjacent. In addition, occurring sparse matrices are converted into permuted direct sums or Kronecker products, if possible, to obtain a concise representation. As an example, consider the sparse matrix

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

M can be permuted to be a direct sum of DFT_2s

$$[(2, 3, 5, 4, 6), 6] \cdot (\mathrm{DFT}_2 \oplus \mathrm{DFT}_2 \oplus \mathrm{DFT}_2) \cdot [(2, 6, 4, 5, 3), 6]$$

or, even more concisely

$$[(2, 3, 5, 4, 6), 6] \cdot (\mathbf{1}_3 \otimes \mathrm{DFT}_2) \cdot [(2, 6, 4, 5, 3), 6].$$

Note that it is an easy task to convert this expression into the original matrix above. In the same way, all permutations in the following expressions (apart from the first or the last one) can be removed leaving a product of sparse matrices each of which can be computed in place. AREP also recognizes scrambled rotation matrices and makes this property explicit. For example, a matrix of the form

$$R = \begin{bmatrix} -\sin\alpha & \cos\alpha \\ \cos\alpha & \sin\alpha \end{bmatrix}$$

would be transformed into the expression

$$[(1, 2), 2] \cdot \mathrm{R}_\alpha.$$

Every matrix expression represents an algorithm for performing a matrix-vector multiplication. The number of multiplications and additions/subtractions required by this algorithm can easily be determined. For example, the matrices

$$\mathrm{DFT}_2,\ \mathbf{1}_2 \otimes \mathrm{DFT}_2,\ \mathbf{1}_2 \oplus \mathrm{DFT}_2,\ \tfrac{1}{2} \cdot \mathrm{DFT}_2$$

require two additions, four additions, two additions, two multiplications, and two additions, respectively. Multiplications with -1 are not counted. Rotation matrices R_α, and scalar multiples thereof, are thought of being realized with three multiplications and three additions, according to the known factorization

$$\begin{bmatrix} x & y \\ -y & x \end{bmatrix} = \begin{bmatrix} 1 & 1 & 0 \\ 0 & -1 & 1 \end{bmatrix} \cdot \begin{bmatrix} x-y & 0 & 0 \\ 0 & y & 0 \\ 0 & 0 & x+y \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 \\ 1 & 1 \\ 0 & 1 \end{bmatrix}.$$

The definitions of the transforms considered follow [38]. A matrix M representing a transform is always applied from the left $x \mapsto M \cdot x$. The runtime for generating the algorithms, i.e., matrix factorizations, was in all cases less than 40 s CPU time on a 233 MHz Pentium II, with 128 MB RAM, running Linux 2.0.36.

A. DFT: Cooley–Tukey

Algorithm 1 finds the Cooley–Tukey factorization of DFT_n, as illustrated in Example 3 for $n = 4$.

B. Cyclic Convolution

Algorithm 1 finds the factorization of an $(n \times n)$ circulant matrix into two DFT_ns as illustrated for $n = 4$ in Example 4. This represents a cyclic convolution.

C. DFT: Rader

The Rader FFT [4] computes a DFT_p of prime size p using two DFTs of size $p-1$. We apply Algorithm 1 to the case $n = 5$ and find the perm-perm symmetry

$$\phi_1\colon x \mapsto [(2, 3, 5, 4), 4]$$
$$\phi_2\colon x \mapsto [(2, 4, 5, 3), 4]$$

with cyclic symmetry group $\mathrm{Z}_4 = \langle x | x^4 = 1 \rangle$. In other words, the permutation $(2, 3, 5, 4)$ in the time-domain corresponds to the permutation $(2, 4, 5, 3)$ in the frequency-domain. The symmetry leads to the Rader factorization

$$\begin{aligned} \mathrm{DFT}_5 = {} & [(4,5),5] \\ & \cdot (\mathbf{1}_1 \oplus ((\mathrm{DFT}_2 \otimes \mathbf{1}_2) \cdot \mathrm{diag}(1,1,1,\omega_4) \cdot (\mathbf{1}_2 \otimes \mathrm{DFT}_2))) \\ & \cdot [(1,4)(2,5,3),(a,b,c,1,1)] \\ & \cdot \left(\mathbf{1}_3 \oplus \begin{bmatrix} 1 & 4 \\ 1 & -1 \end{bmatrix}\right) \cdot [(1,4)(2,3,5),5] \\ & \cdot (\mathbf{1}_1 \oplus \tfrac{1}{4} \cdot (\mathbf{1}_2 \otimes \mathrm{DFT}_2) \cdot \mathrm{diag}(1,1,1,-\omega_4) \\ & \quad \cdot (\mathrm{DFT}_2 \otimes \mathbf{1}_2)) \cdot [(3,4,5),5]. \end{aligned}$$

The first two lines contain the matrix A_1 (essentially a DFT_4), the last two lines the matrix A_2^{-1} (essentially an inverse DFT_4), and the middle two lines contain the matrix D from Algorithm 1. (a, b, c are complex constants whose actual value has been omitted for the sake of clarity.)

D. DCT, Type II, and III

The discrete cosine transform of type III, $\mathrm{DCT}^{(\mathrm{III})}$, is defined as the matrix

$$\mathrm{DCT}_n^{(\mathrm{III})} = \left[\sqrt{\tfrac{2}{n}} \cdot a_\ell \cdot \cos\tfrac{(2k+1)\ell\pi}{2n} \,\middle|\, k, \ell = 0 \ldots n-1\right]$$

where $a_k = 1/\sqrt{2}$ for $k = 0$ and $a_k = 1$ elsewhere. $\mathrm{DCT}^{(\mathrm{II})}$ is the transpose of $\mathrm{DCT}^{(\mathrm{III})}$. We compute a perm-irred symmetry for $\mathrm{DCT}_8^{(\mathrm{III})}$ with dihedral symmetry group $G = \mathrm{D}_{16} = \langle x, y | x^8 = y^2 = 1,\ y^{-1}xy = x^{-1} \rangle$ and representations

$$\phi_1\colon x \mapsto [(1, 3, 5, 7, 8, 6, 4, 2), 8],$$
$$y \mapsto [(2, 3)(4, 5)(6, 7), 8]$$
$$\phi_2\colon x \mapsto M_1,\ y \mapsto M_2$$

where, using $c_k = \cos k\pi/8$ and $s_k = \sin k\pi/8$

$$M_1 = \begin{bmatrix} c_0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & c_2 & 0 & 0 & 0 & 0 & 0 & s_2 \\ 0 & 0 & c_4 & 0 & 0 & 0 & s_4 & 0 \\ 0 & 0 & 0 & c_6 & 0 & s_6 & 0 & 0 \\ 0 & 0 & 0 & 0 & c_8 & 0 & 0 & 0 \\ 0 & 0 & 0 & s_{10} & 0 & c_{10} & 0 & 0 \\ 0 & 0 & s_{12} & 0 & 0 & 0 & c_{12} & 0 \\ 0 & s_{14} & 0 & 0 & 0 & 0 & 0 & c_{14} \end{bmatrix}$$

$$M_2 = \begin{bmatrix} c_0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & c_1 & 0 & 0 & 0 & 0 & 0 & s_1 \\ 0 & 0 & c_2 & 0 & 0 & 0 & s_2 & 0 \\ 0 & 0 & 0 & c_3 & 0 & s_3 & 0 & 0 \\ 0 & 0 & 0 & 0 & s_4 & 0 & 0 & 0 \\ 0 & 0 & 0 & s_5 & 0 & c_5 & 0 & 0 \\ 0 & 0 & s_6 & 0 & 0 & 0 & c_6 & 0 \\ 0 & s_7 & 0 & 0 & 0 & 0 & 0 & c_7 \end{bmatrix}.$$

Since $\mathrm{DCT}^{(\mathrm{III})} \cdot x$ is equivalent to $x^T \cdot \mathrm{DCT}^{(\mathrm{II})}$, we get the following interpretation for the $\mathrm{DCT}^{(\mathrm{II})}$. Permuting with $(1, 3, 5, 7, 8, 6, 4, 2)$ or $(2, 3)(4, 5)(6, 7)$ in the time-domain corresponds to multiplication with M_1 or M_2, respectively, in the frequency domain.

The symmetry leads to the following factorization of $\mathrm{DCT}_8^{(\mathrm{III})}$, which already has been derived by Minkwitz using a preliminary version of Algorithm 1, as sketched in [27] and [28]. By symbolic transposition (the order of the product is reversed and each factor transposed using mathematical properties), we get a factorization of $\mathrm{DCT}_8^{(\mathrm{II})}$, which essentially corresponds to the first algorithm found by Chen *et al.* [11], which directly computes the $\mathrm{DCT}^{(\mathrm{II})}$ without using the DFT.

$$\begin{aligned} \mathrm{DCT}_8^{(\mathrm{III})} &= [(1, 2, 6, 8)(3, 7, 5, 4), 8] \\ &\cdot (\mathbf{1}_2 \otimes ((\mathbf{1}_2 \otimes \mathrm{DFT}_2) \cdot [(2, 3), 4] \cdot (\mathrm{DFT}_2 \oplus \mathbf{1}_2))) \\ &\cdot [(2, 7, 6, 8, 5, 4, 3), 8] \cdot \left(\mathbf{1}_4 \oplus \tfrac{1}{\sqrt{2}} \cdot \mathrm{DFT}_2 \oplus \mathbf{1}_2\right) \\ &\cdot [(5, 6), 8] \cdot ((\mathrm{DFT}_2 \otimes \mathbf{1}_3) \oplus \mathbf{1}_2) \cdot [(2, 8, 3, 7, 4), 8] \\ &\cdot \tfrac{1}{2} \cdot \left(\tfrac{1}{\sqrt{2}} \cdot \mathbf{1}_2 \oplus \mathrm{R}_{(13/8)\pi} \oplus \mathrm{R}_{(17/16)\pi} \oplus \mathrm{R}_{(11/16)\pi}\right) \\ &\cdot [(2, 5)(4, 7)(6, 8), 8] \\ &\qquad (13 \text{ mults}, 29 \text{ adds}) \end{aligned}$$

and, by transposition

$$\begin{aligned} \mathrm{DCT}_8^{(\mathrm{II})} &= [(2, 5)(4, 7)(6, 8), 8] \\ &\cdot \tfrac{1}{2} \cdot \left(\tfrac{1}{\sqrt{2}} \cdot \mathbf{1}_2 \oplus \mathrm{R}_{(3/8)\pi} \oplus \mathrm{R}_{(15/16)\pi} \oplus \mathrm{R}_{(21/16)\pi}\right) \\ &\cdot [(2, 4, 7, 3, 8), 8] \cdot ((\mathrm{DFT}_2 \otimes \mathbf{1}_3) \oplus \mathbf{1}_2) \cdot [(5, 6), 8] \\ &\cdot \left(\mathbf{1}_4 \oplus \tfrac{1}{\sqrt{2}} \cdot \mathrm{DFT}_2 \oplus \mathbf{1}_2\right) \cdot [(2, 3, 4, 5, 8, 6, 7), 8] \\ &\cdot (\mathbf{1}_2 \otimes ((\mathrm{DFT}_2 \oplus \mathbf{1}_2) \cdot [(2, 3), 4] \cdot (\mathbf{1}_2 \otimes \mathrm{DFT}_2))) \\ &\cdot [(1, 8, 6, 2)(3, 4, 5, 7), 8] \\ &\qquad (13 \text{ mults}, 29 \text{ adds}). \end{aligned}$$

Looking at the factorization of $\mathrm{DCT}_8^{(\mathrm{III})}$, the first four lines give the matrix A_1 from Algorithm 1, the last line contains the permutation matrix A_2^{-1} (which makes the block structure of M_1 and M_2 explicit), and the fifth line gives the matrix D.

The algorithms for $\mathrm{DCT}_8^{(\mathrm{III})}$ and $\mathrm{DCT}_8^{(\mathrm{II})}$ have the same arithmetic cost as the best known algorithms [8], [11], [12], [15], [17]–[19], [39]. Note that those who use only 12 multiplications do not normalize the first row of the $\mathrm{DCT}_8^{(\mathrm{II})}$, which saves one multiplication. The only algorithm that claims 11 multiplications [40] considers a scaled version of the $\mathrm{DCT}^{(\mathrm{II})}$ matrix

$$\mathrm{DCT}_n^{(\mathrm{II})'} = \sqrt{2} \cdot \sqrt{\tfrac{n}{2}} \cdot \mathrm{DCT}_n^{(\mathrm{II})}.$$

Multiplying by scalars conserves the perm-irred symmetry (it just changes the matrix D in Algorithm 1) and AREP also finds a factorization with 11 multiplications

$$\begin{aligned} \mathrm{DCT}_8^{(\mathrm{II})'} &= [(2, 5)(4, 7)(6, 8), 8] \\ &\cdot \left(\mathbf{1}_2 \oplus \sqrt{2} \cdot \mathrm{R}_{(3/8)\pi} \oplus \sqrt{2} \cdot \mathrm{R}_{(15/16)\pi} \oplus \sqrt{2} \cdot \mathrm{R}_{(21/16)\pi}\right) \\ &\cdot [(2, 4, 7, 3, 8), 8] \cdot ((\mathrm{DFT}_2 \otimes \mathbf{1}_3) \oplus \mathbf{1}_2) \cdot [(5, 6), 8] \\ &\cdot \left(\mathbf{1}_4 \oplus \tfrac{1}{\sqrt{2}} \cdot \mathrm{DFT}_2 \oplus \mathbf{1}_2\right) \cdot [(2, 3, 4, 5, 8, 6, 7), 8] \\ &\cdot (\mathbf{1}_2 \otimes ((\mathrm{DFT}_2 \oplus \mathbf{1}_2) \cdot [(2, 3), 4] \cdot (\mathbf{1}_2 \otimes \mathrm{DFT}_2))) \\ &\cdot [(1, 8, 6, 2)(3, 4, 5, 7), 8] \\ &\qquad (11 \text{ mults}, 29 \text{ adds}). \end{aligned}$$

We want to mention that the $\mathrm{DCT}^{(\mathrm{III})}$ (and, hence, the $\mathrm{DCT}^{(\mathrm{II})}$) also has a mon-mon symmetry. For example, for the case $n = 8$, the symmetry group is the direct product $\mathrm{Z}_2 \times \mathrm{Z}_8$. In fact, this symmetry has been used by Feig and Winograd to derive a fast $\mathrm{DCT}^{(\mathrm{II})}$ algorithm [19] and a lower bound for the number of nonrational multiplications necessary for computing the $\mathrm{DCT}^{(\mathrm{II})}$ (for $n = 8$ the optimal number is 11, where the first row of the $\mathrm{DCT}^{(\mathrm{II})}$ is unscaled). They essentially follow Algorithm 1 with the difference that ϕ_1 and ϕ_2 are only decomposed over the rational numbers $\mathbb{Q}$ (which yields a coarser decomposition of ϕ_1 and ϕ_2) using rational matrices A_1 and A_2. All nonrational multiplications then are concentrated in the block diagonal matrix D. AREP currently is only capable to decompose representations over $\mathbb{C}$.

E. DCT, Type IV

The discrete cosine transform of type IV, $\mathrm{DCT}^{(\mathrm{IV})}$, is defined as the matrix

$$\mathrm{DCT}_n^{(\mathrm{IV})} = \left[\sqrt{\tfrac{2}{n}} \cdot \cos \tfrac{(2k+1)(2\ell+1)\pi}{4n} | k, \ell = 0 \ldots n-1\right].$$

We compute a mon-irred symmetry for $\mathrm{DCT}_8^{(\mathrm{IV})}$ with dihedral symmetry group $G = \mathrm{D}_{32} = \langle x, y | x^{16} = y^2 = 1,\ y^{-1}xy = x^{-1} \rangle$ and representations

$$\begin{aligned} \phi_1 &: x \mapsto [(1, 3, 5, 7, 8, 6, 4, 2), (1, 1, 1, 1, 1, 1, 1, -1)] \\ &\quad y \mapsto [(2, 3)(4, 5)(6, 7), (1, 1, 1, 1, 1, 1, 1, -1)] \\ \phi_2 &: x \mapsto M_1,\ y \mapsto M_2 \end{aligned}$$

i.e., compared to the perm-irred symmetry of the $\mathrm{DCT}_8^{(\mathrm{III})}$, the last column of the images of $\phi_1(x)$ and $\phi_1(y)$ are mul-

tiplied by -1, which also leads to a larger group. Using $c_k = \cos k\pi/16$, $s_k = \sin k\pi/16$, the matrices M_1, M_2 are given by

$$M_1 = \begin{bmatrix} c_2 & 0 & 0 & 0 & 0 & 0 & 0 & s_2 \\ 0 & c_6 & 0 & 0 & 0 & 0 & s_6 & 0 \\ 0 & 0 & c_{10} & 0 & 0 & s_{10} & 0 & 0 \\ 0 & 0 & 0 & c_{14} & s_{14} & 0 & 0 & 0 \\ 0 & 0 & 0 & s_{18} & c_{18} & 0 & 0 & 0 \\ 0 & 0 & s_{22} & 0 & 0 & c_{22} & 0 & 0 \\ 0 & s_{26} & 0 & 0 & 0 & 0 & c_{26} & 0 \\ s_{30} & 0 & 0 & 0 & 0 & 0 & 0 & c_{30} \end{bmatrix}$$

$$M_2 = \begin{bmatrix} c_1 & 0 & 0 & 0 & 0 & 0 & 0 & s_1 \\ 0 & c_3 & 0 & 0 & 0 & 0 & s_3 & 0 \\ 0 & 0 & c_5 & 0 & 0 & s_5 & 0 & 0 \\ 0 & 0 & 0 & c_7 & s_7 & 0 & 0 & 0 \\ 0 & 0 & 0 & s_9 & c_9 & 0 & 0 & 0 \\ 0 & 0 & s_{11} & 0 & 0 & c_{11} & 0 & 0 \\ 0 & s_{13} & 0 & 0 & 0 & 0 & c_{13} & 0 \\ s_{15} & 0 & 0 & 0 & 0 & 0 & 0 & c_{15} \end{bmatrix}.$$

The symmetry leads to the following factorization of $\mathrm{DCT}_8^{(\mathrm{IV})}$. Since $\mathrm{DCT}^{(\mathrm{IV})}$ is symmetric, transposition leads to another factorization of $\mathrm{DCT}_8^{(\mathrm{IV})}$, which is very close to the fast algorithm given by Wang [12].

$$\begin{aligned} \mathrm{DCT}_8^{(\mathrm{IV})} = {} & [(1,2,8)(3,6,5),(1,-1,1,1,1,-1,1,1)] \\ & \cdot \left(\mathbf{1}_2 \otimes \left(\left(\mathbf{1}_2 \oplus \tfrac{1}{\sqrt{2}} \cdot \mathrm{DFT}_2\right) \cdot [(3,4),4]\right.\right. \\ & \qquad \left.\left. \cdot\ (\mathrm{DFT}_2 \otimes \mathbf{1}_2)\right)\right) \\ & \cdot [(1,3)(2,4)(5,7)(6,8),8] \\ & \cdot (\mathbf{1}_4 \oplus \mathrm{R}_{(15/8)\pi} \oplus \mathrm{R}_{(11/8)\pi}) \\ & \cdot (\mathrm{DFT}_2 \otimes \mathbf{1}_4) \cdot [(3,5,7)(4,6,8),8] \\ & \cdot \tfrac{1}{2} \cdot (\mathrm{R}_{(31/32)\pi} \oplus \mathrm{R}_{(19/32)\pi} \oplus \mathrm{R}_{(27/32)\pi} \oplus \mathrm{R}_{(23/32)\pi}) \\ & \cdot [(1,8,5,6,3,2)(4,7),8] \\ & \qquad (20 \text{ mults},\ 38 \text{ adds}). \end{aligned}$$

The first six lines correspond to the decomposition matrix A_1 of ϕ_1 in Algorithm 1, the seventh line to D, and the last line contains the permutation matrix A_2^{-1}, the inverse of which permutes ϕ_2 to be a direct sum.

An algorithm with two additions less can be found in [17].

F. DST, Type II, and III

The discrete sine transform of type III, $\mathrm{DST}^{(\mathrm{III})}$, is defined as the matrix

$$\mathrm{DST}_n^{(\mathrm{III})} = \left[\sqrt{\tfrac{2}{n}} \cdot a_k \cdot \sin \tfrac{(2k+1)(\ell+1)\pi}{2n} | k, \ell = 0..n-1\right]$$

where $a_k = 1/\sqrt{2}$ for $k = 0$ and $a_k = 1$ elsewhere. $\mathrm{DST}^{(\mathrm{II})}$ is the transpose of $\mathrm{DST}^{(\mathrm{III})}$. We compute a perm-irred symmetry for $\mathrm{DST}_8^{(\mathrm{III})}$ with dihedral symmetry group $G = \mathtt{D}_{16} = \langle x, y | x^8 = y^2 = 1, y^{-1}xy = x^{-1}\rangle$ and representations

$$\begin{aligned} \phi_1 &: x \mapsto [(1,3,5,7,8,6,4,2),(-1,1,1,1,1,1,1,-1)] \\ &\quad y \mapsto [(2,3)(4,5)(6,7),(-1,1,1,1,1,1,1,-1)] \\ \phi_2 &: x \mapsto M_1,\ y \mapsto M_2 \end{aligned}$$

i.e., compared to the perm-irred symmetry of the $\mathrm{DCT}_8^{(\mathrm{III})}$, the first and last column of the images of $\phi_1(x)$ and $\phi_1(y)$ are multiplied by -1. The matrices M_1 and M_2 (not given due to lack of space) have entries $\neq 0$ only on the diagonal and at positions (i, j) with $i + j = 8$, $i, j = 1 \ldots 8$.

The symmetry leads to a factorization of $\mathrm{DST}_8^{(\mathrm{III})}$ and, hence, to a factorization of $\mathrm{DST}_8^{(\mathrm{II})}$, which requires 13 additions and 29 multiplications.

Since $\mathrm{DCT}^{(\mathrm{II})}$, $\mathrm{DCT}^{(\mathrm{III})}$, $\mathrm{DST}^{(\mathrm{II})}$, and $\mathrm{DST}^{(\mathrm{III})}$ all have the same arithmetic cost (because type II and III are transposed and [17, Sec. 4.2]), the algorithms found by AREP are among the best known algorithms.

G. DST, Type IV

The discrete sine transform of type IV $\mathrm{DST}^{(\mathrm{IV})}$ is defined as the matrix

$$\mathrm{DST}_n^{(\mathrm{IV})} = \left[\sqrt{\tfrac{2}{n}} \cdot \sin \tfrac{(2k+1)(2\ell+1)\pi}{4n} | k, \ell = 0 \ldots n-1\right].$$

We compute a mon-irred symmetry for $\mathrm{DCT}_8^{(\mathrm{IV})}$ with dihedral symmetry group $G = \mathtt{D}_{32} = \langle x, y | x^{16} = y^2 = 1,\ y^{-1}xy = x^{-1}\rangle$ and representations

$$\begin{aligned} \phi_1 &: x \mapsto [(1,3,5,7,8,6,4,2),(-1,1,1,1,1,1,1,1)] \\ &\quad y \mapsto [(2,3)(4,5)(6,7),(-1,1,1,1,1,1,1,1)] \\ \phi_2 &: x \mapsto M_1,\ y \mapsto M_2 \end{aligned}$$

i.e., the difference to the perm-irred symmetry of the $\mathrm{DCT}_8^{(\mathrm{III})}$ lies only in the first column of the images being multiplied by -1. The matrices M_1 and M_2 (not given due to lack of space) have entries $\neq$ only on the diagonal and the opposite diagonal.

The symmetry leads to a factorization of $\mathrm{DST}_8^{(\mathrm{IV})}$, which requires 20 multiplications and 38 additions. As for the $\mathrm{DCT}^{(\mathrm{IV})}$, this is two additions more as in the best known algorithm [17].

H. DCT and DST, Type I

Although the transforms $\mathrm{DCT}^{(\mathrm{I})}$ and $\mathrm{DST}^{(\mathrm{I})}$ do not have a mon-irred symmetry, they do possess a mon-mon symmetry that can be used for their factorization. However, the algorithms obtained this way are not as good as those from [17].

I. Hartley Transform

The discrete Hartley transform DHT_n is defined as the matrix

$$\mathrm{DHT}_n = \left[\cos \tfrac{2k\ell\pi}{n} + \sin \tfrac{2k\ell\pi}{n} | k, \ell = 0 \ldots n-1\right].$$

Note that we omitted the normalization factor $1/\sqrt{n}$ in the definition to obtain a fair comparison to known algorithms. The DHT_8 has a perm-irred symmetry with dihedral symmetry

group $G = \mathtt{D}_{16} = \langle x, y | x^8 = y^2 = 1, y^{-1}xy = x^{-1} \rangle$ and representations

$$\begin{aligned}\phi_1 &: x \mapsto [(1,2,3,4,5,6,7,8),8]\\ &\quad y \mapsto [(2,8)(3,7)(4,6),8]\\ \phi_2 &: x \mapsto M_1,\ y \mapsto [(2,8)(3,7)(4,6),8].\end{aligned}$$

The only nonzero entries of the matrix M_1 are on the diagonal and at (i, j), where $i + j = 9$. The symmetry yields the following factorization:

$$\begin{aligned}\mathrm{DHT}_8 = {} & [(1,8)(2,4)(3,5,7,6),8]\\ & \cdot (\mathbf{1}_2 \otimes ((\mathbf{1}_2 \otimes \mathrm{DFT}_2) \cdot [(2,3),4] \cdot (\mathrm{DFT}_2 \oplus \mathbf{1}_2)))\\ & \cdot [(2,7,6,8,5,4,3),8] \cdot \left(\mathbf{1}_4 \oplus -\tfrac{1}{\sqrt{2}} \cdot \mathrm{DFT}_2 \oplus \mathbf{1}_2\right)\\ & \cdot ((\mathrm{DFT}_2 \otimes \mathbf{1}_3) \oplus \mathbf{1}_2)\\ & \cdot \Big[(2,5,3,6,4)(7,8)\\ & \qquad \left(1, -1, -\sqrt{2}, -\sqrt{2}, \sqrt{2}, \sqrt{2}, 1, 1\right)\Big]\\ & \cdot (\mathbf{1}_6 \oplus -\mathrm{DFT}_2) \cdot [(2,5,8,7,3,4),8]\\ & \qquad (6 \text{ mults}, 22 \text{ adds}).\end{aligned}$$

Closer investigation shows that two of the $\sqrt{2}$ could be canceled against the $1/\sqrt{2}$, yielding four multiplications less.

The best algorithm for the DHT_8 with respect to arithmetic operations seems to be the split-radix algorithm given in Sorensen *et al.* [41] and needs two multiplications and 22 additions.

J. Haar Transform

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

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

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

for $k \geq 1$. The transpose of the Haar transform has a perm-irred symmetry. The symmetry group is an iterated wreath product [26]. For $k = 3$, we obtain the following factorization:

$$\begin{aligned}\mathrm{HT}_8 = {} & \Big[(3,4)(5,7)(6,8)\\ & \left(\tfrac{1}{8}, -\tfrac{1}{8}, \tfrac{1}{4\sqrt{2}}, \tfrac{1}{4\sqrt{2}}, \tfrac{1}{4}, \tfrac{1}{4}, \tfrac{1}{4}, \tfrac{1}{4}\right)\Big]\\ & \cdot (\mathrm{DFT}_2 \oplus \mathbf{1}_6) \cdot [(2,5,3)(4,6),8]\\ & \cdot (\mathbf{1}_2 \otimes ((\mathrm{DFT}_2 \oplus \mathbf{1}_2) \cdot [(2,3),4] \cdot (\mathbf{1}_2 \otimes \mathrm{DFT}_2)))\\ & \cdot [(1,5)(2,6)(3,7)(4,8),8]\\ & \qquad (8 \text{ mults}, 14 \text{ adds}).\end{aligned}$$

The first two lines contain the matrix D from Algorithm 1 and the other lines the matrix A_1 (we decomposed the transpose of HT(8) and transposed the result). The number of operations coincides with the best known algorithm [42].

K. Wreath Product Transform

In a recent paper, [26], decomposition matrices for permutation representations of certain groups, which are called iterated wreath products, have been proposed for image processing. By construction, these transforms possess symmetry in our definition. Thus, we readily obtain the following factorization of the (16×16) transform W_{16} given in [26, p. 117] (where it is called A; we want to note that this transform has an error in column 15, where the last $-i$ should read i). For brevity, we denote $j = \omega_4$.

$$\begin{aligned}\mathrm{W}_{16} = {} & [(1,14,7)(2,15,8)(3,16,5)(4,13,6)(9,11)\\ & \quad (10,12),16]\\ & \cdot (\mathbf{1}_2 \otimes ((\mathbf{1}_2 \otimes ((\mathrm{DFT}_2 \otimes \mathbf{1}_2) \cdot \mathrm{diag}(1,1,1,j)\\ & \qquad \cdot (\mathbf{1}_2 \otimes \mathrm{DFT}_2)))\\ & \cdot [(2,5)(4,7),8] \cdot (\mathrm{DFT}_2 \oplus \mathbf{1}_6)))\\ & \cdot [(3,13,11,15,7,5,9)(4,14,12,16,8,6,10),\\ & \quad (1,1,1,j,1,1,1,1,1,1,1,1,1,1,1)]\\ & \cdot ((\mathrm{DFT}_2 \otimes \mathbf{1}_2) \oplus \mathbf{1}_{12})\\ & \cdot [(5,14,8,13,6,16)(7,15)(9,10,12)\\ & \quad (1,j,-1,-j,-j,-1,-1,-j,-1,1,1,-1,j\\ & \qquad -1,-1,j)]\\ & \qquad (11 \text{ mults by } j,\ 24 \text{ adds}).\end{aligned}$$

VI. Conclusion and Future Research

We have presented an entirely automatic method for symbolically deriving fast algorithms for an important class of discrete linear signal transforms. This class includes the DFT, DCT, DST, Hartley, and Haar transforms. In most cases, the derived algorithms were among the best ones known. The approach is based on the definition of "symmetry" of a transform as a pair of group representations, which operate in the time-domain and the frequency-domain, respectively, leaving the transform invariant. More precisely, the considered transform matrices can be factorized because they are decomposition matrices of monomial group representations.

The results of this paper open at least the following two research questions.

1) How do symmetry and signal processing properties of a transform relate to each other?
2) Is it possible to extend the approach described to derive a larger class of fast signal transforms?

AREP includes an interface to SPL. SPL is a domain specific language and compiler for 1) representing a fast signal transform given as a matrix expression like the ones generated by AREP and 2) translate it into an efficient, e.g., C program for computing the transform. SPL is under development within [43]. The interface between AREP and SPL allows the automatic implementation of all algorithms derived by AREP.

Appendix

In the following, we will give an overview on Step 1 and 2 of Algorithm 1. For a comprehensive treatment including all technical details, see [27] and [29]–[33], which also provide the necessary prerequisites from representation theory of finite groups.

Of particular interest might be the recursion formula for Step 2, given in Theorem 2, which gives the explanation for the algebraic structure of the fast transforms found in Section V.

A. Algorithm 1, Step 1

In Step 1 of Algorithm 1, the mon-mon or mon-irred symmetry of the matrix M has to be computed. The two types require different approaches.

For the mon-mon symmetry, we are interested in the group G of all pairs (L, R) of monomial matrices such that $LM = MR$. (The representations ϕ_1 and ϕ_2 of Section III are the projections $(L, R) \mapsto L$ and $(L, R) \mapsto R$.) However, this general definition may lead to an infinite G. Therefore, we artificially restrict L and R to contain kth roots of unity. Let $\mathrm{MonMon}_k(M)$ denote the group of all pairs of monomial matrices (L, R) such that $LM = MR$ and all entries in L and R are kth roots of unity.

To construct $\mathrm{MonMon}_k(M)$, we replace each entry M_{ij} with the $(k \times k)$-matrix $M' = [M_{ij}\omega_k^{r+s} | r, s]$, where ω_k denotes a fixed primitive kth root of unity. This encoding of M turns monomial operations on M (with kth roots of unity) into permutations operations on the larger matrix M'. The parameter k is chosen dependent on the entries of M. The encoding has first been described in [44] for the special case of finite fields. It reduces MonMon_k to the well-known problem of constructing all pairs (L, R) of permutations such that $LM' = M'R$ for a given matrix M' (the perm-perm symmetry of M'). The most efficient methods to solve the latter problem are partition-based backtracking methods that systematically try all permutations, removing entire branches quickly. This is described by Leon in [44], who also distributes a very efficient C-program to compute the symmetry. Moreover, [29] describes a program in GAP to compute MonMon_k, and it is proven in [30] and [31] that MonMon_k can indeed be computed the way we have claimed.

Now, we turn to the other important type of symmetry: the mon-irred symmetry. Here, we are interested in all monomial matrices L such that $R = MLM^{-1}$ is permuted block-diagonal for invertible M. Formally, "permuted block-diagonal" means that there is a permutation π such that $R = \pi^{-1} \cdot (R_1, \oplus \ldots \oplus R_m) \cdot \pi$ for smaller matrices $R_1, \ldots, R_m$. As a quantitative measure for block-diagonality, we define the *conjugated block structure* (cbs) of a matrix A as the partition $\mathrm{cbs}(A) = \{1, \ldots, n\}/\sim^*$, where $\sim^*$ is the reflexive-symmetric-transitive closure of the relation $\sim$ defined by $i \sim j \Leftrightarrow A_{ij} \neq 0$. The partitions are partially ordered by the refinement.

There are two approaches to the mon-irred symmetry. The first one essentially enumerates all monomial matrices L and collects them into groups according to the block structures $\mathrm{cbs}(MLM^{-1})$ to which they give rise. The result is a list of pairs (G, p) such that G is the group of all monomial matrices L such that $\mathrm{cbs}(MLM^{-1}) \leq p$ and p is the join of all $\mathrm{cbs}(MLM^{-1})$. Each of the groups G qualifies as a mon-irred symmetry; it only has to be tested if the matrices $R = MLM^{-1}$ form a representation of G that is a direct sum of irreducible representations.

The second approach to the mon-irred symmetry essentially enumerates all partitions p and constructs the group G of all matrices L such that $\mathrm{cbs}(MLM^{-1})$ is a refinement of p. The result is the same list of pairs (G, p) as before. The main difference to the first method is that one can restrict the enumeration of partitions to the cases that will be most useful for the decomposition of the signal transform, namely, those with many small blocks. This is much faster than running through all monomial matrices. For details on the methods and on a number of important improvements, see [29] and [32]. All of the mon-irred symmetries used for the examples in this paper are found in a few seconds on a standard workstation using our implementation in GAP for the library AREP.

B. Algorithm 1, Step 2

Step 2 of Algorithm 1 requires to decompose a given monomial representation ϕ of a group G into a direct sum of irreducible representations. In addition, the corresponding decomposition matrix has to be computed as a product of structured sparse matrices.

First, we need to introduce the notion of a *transitive* monomial representation. Let $n = \deg \phi$, and denote by e_i the ith canonical base vector (ith entry $= 1$, $= 0$ else). Then, ϕ is called transitive if for all i, j there is a $g \in G$ such that (the monomial matrix) $\phi(g)$ maps e_i to a multiple of e_j.

The key construct for the decomposition algorithm is the *induction* of representations. In short, induction constructs a representation of G from a representation of a subgroup of G. More precisely, let $H \leq G$ be a subgroup with representation λ, and let $T = (t_1, \ldots, t_k)$ be a transversal (i.e., a system of representatives of the right cosets of H in G). Then

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

where $\dot{\phi}(x) = \phi(x)$ for $x \in H$ and the all-zero matrix else is called the induction of ϕ to G with transversal T. Note that $(\phi \uparrow_T G)(g)$ is a block-permuted matrix, i.e., for all $i = 1 \ldots k$, there is exactly one j with $t_i g t_j^{-1} \in H$. If λ has degree one, then the induction is monomial.

Finally, recall that a group G is called solvable if there is a sequence $N_i \leq G$, $i = 1 \ldots r$ of subgroups, such that

$$\{1\} = N_1 \leq N_2 \leq \cdots \leq N_r = G$$

and N_i is normal of prime index in N_{i+1} for $i = 1 \ldots r - 1$.

Now, we can formulate a coarse version of the recursive decomposition algorithm. The algorithm essentially conjugates a monomial representation to be an induction, which is decomposed along a chain of normal subgroups using a recursion formula for decomposition matrices. For a complete version of this algorithm, including the underlying theorems, see [33].

Algorithm 2: Given a monomial representation ϕ of a solvable group G. ϕ shall be decomposed, i.e.,

$$\phi^A = \rho_1 \oplus \cdots \oplus \rho_m$$

where all ρ_i are irreducible, and A is a product of structured sparse matrices.

Case 1: ϕ is not transitive.

1) Decompose ϕ with a permutation P into a direct sum of transitive representations $\phi^P = \phi_1 \oplus \cdots \oplus \phi_\ell$.

2) Recurse with ϕ_i, $i = 1 \cdots \ell$ to obtain decomposition matrices B_i.

$A = P \cdot (B_1 \oplus \cdots \oplus B_\ell)$ is a decomposition matrix for ϕ.

Case 2: ϕ is transitive.

1) Decompose ϕ with a diagonal matrix D into an induction $\lambda \uparrow_T G$, where λ has degree 1.
2) Recurse with $\lambda \uparrow_T G$ to obtain a decomposition matrix B.

$A = D \cdot B$ is a decomposition matrix for ϕ.

Case 3: $\phi = \lambda \uparrow_T G$, where λ is a representation of $H \leq G$, and there exists a normal subgroup $H \leq N \leq G$ of prime index p in G.

1) Decompose ϕ with a monomial matrix M into a double induction $\phi^M = (\lambda \uparrow_{T_1} N) \uparrow_{T_2} G$.
2) Recurse with $\lambda \uparrow_{T_1} N$ to obtain a decomposition matrix B.

$M \cdot A$ is a decomposition matrix for ϕ, where A is given by Theorem 2.

Case 4: $\phi = \lambda \uparrow_T G$, λ is a representation of $H \leq G$, and there exists a normal subgroup $H \nleq N \leq G$ of prime index p in G.

We omit this step.

Note that at least one of the cases always applies since G is solvable. We omitted Case 4 since it did not play a role for the examples considered in this paper. The recursion formula for Case 3 is given in Theorem 2. We may only want to look at the actual formula for A, omitting the technical details. Obviously, all factors are sparse. For the special case $G = \mathbb{Z}_n = \langle x | x^n = 1 \rangle$ and $\phi_1 \colon x \mapsto [(1, 2, \ldots, n), n]$, the permutation P vanishes, $d = 0$, and the formula reduces exactly to the Cooley–Tukey FFT.

Theorem 2: Let $N \leq G$ be a normal subgroup of prime index p with transversal $T = (t^0, t^1, \ldots, t^{(p-1)})$. Assume ϕ is a representation of N of degree n with decomposition matrix B such that $\phi^B = \bigoplus_{i=1}^{k} \rho_i$, where $\rho_1, \ldots, \rho_j$ are exactly those among the ρ_i that have an extension $\overline{\rho}_i$ to G. Denote by $d = \deg(\rho_1) + \cdots + \deg(\rho_j)$ the entire degree of the extensible ρ_i, and set $\overline{\rho} = \overline{\rho}_1 \oplus \cdots \oplus \overline{\rho}_j$. Then, there exists a permutation matrix P such that

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

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

C. Example 4 in Detail

We work out Steps 1 and 2 for Example 4 in greater detail.

Step 1: We choose to compute the mon-mon symmetry of M. Since the matrix M is real, we only consider $\mathrm{MonMon}_2(M)$. The algorithm first constructs the matrix

$$M' = \begin{bmatrix} 1 & -1 & 2 & -2 & 3 & -3 & 4 & -4 \\ -1 & 1 & -2 & 2 & -3 & 3 & -4 & 4 \\ 4 & -4 & 1 & -1 & 2 & -2 & 3 & -3 \\ -4 & 4 & -1 & 1 & -2 & 2 & -3 & 3 \\ 3 & -3 & 4 & -4 & 1 & -1 & 2 & -2 \\ -3 & 3 & -4 & 4 & -1 & 1 & -2 & 2 \\ 2 & -2 & 3 & -3 & 4 & -4 & 1 & -1 \\ -2 & 2 & -3 & 3 & -4 & 4 & -1 & 1 \end{bmatrix}.$$

Any $\{-1, 0, 1\}$-monomial operation on M (permuting and/or negating rows or columns) is a permutation operation on M'. For example, negating the first column of M can be expressed as the exchange of the first two columns of M'. Now, we compute all pairs (L', R') of permutations such that $L'M' = M'R'$ by recursive search. Assume that L' maps row one to itself. Then, R' must map column one to itself as well, …. Assume L' maps row one to row two, then, …, etc. As the result, we find that $L' = R'$ (which is obvious since the diagonal values of M' are different from the rest so they can only be exchanged with each other), and

$$L' \in \langle (1, 3, 5, 7)(2, 4, 6, 8), (1, 2)(3, 4)(5, 6)(7, 8) \rangle.$$

Translating the permutations on M' back into monomial operations on M, we see that the first permutation is just $[(1, 2, 3, 4), 4]$, and the second is a scalar multiplication with -1, which we discard since it does not carry any information about M. Hence, we have found the representations $\phi_1 = \phi_2 \colon x \mapsto [(1, 2, 3, 4), 4]$ of the cyclic group $G = \{1, x, x^2, x^3\}$ as the mon-mon symmetry of M.

Step 2: The remaining task is to decompose the representations $\phi_1 = \phi_2$ using Algorithm 2. Obviously, ϕ_1 is transitive, and we apply Case 2 to find $\phi_1 = 1_E \uparrow_T G$, where 1_E is the trivial representation $1 \mapsto 1$ of the trivial subgroup $E = \{1\} \leq G$, and T is the list $(1, x, x^2, x^3)$ of elements of G. The matrix D is the identity. Next, we apply Case 3 using the subgroup $H = \{1, x^2\} \leq G$ and find $(1_E \uparrow_T G)^{[(2,3),4]} = (1_E \uparrow_{T_1} H) \uparrow_{T_2} G$ with $T_1 = (1, x^2)$ and $T_2 = (1, x)$. Now, we apply Theorem 2. The matrix $B = \mathrm{DFT}_2$ decomposes $(1_E \uparrow_{T_1} H) \colon x^2 \mapsto [(1, 2), 2]$ into $\rho_1 \colon x^2 \mapsto 1$, $\rho_2 \colon x^2 \mapsto -1$. Both representations can be extended to G through $\overline{\rho}_1 \colon x \mapsto 1$, and $\overline{\rho}_2 \colon x \mapsto j$, respectively. In the recursion formula for A, the permutation P vanishes, and evaluating $\overline{\rho} = \overline{\rho}_1 \oplus \overline{\rho}_2$ at the transversal $T_2 = (1, x)$ yields the twiddle factors $\mathrm{diag}(1, 1, 1, j)$. The parameter d equals zero, and we obtain the Cooley–Tukey formula for DFT_4:

$$A = (\mathrm{DFT}_2 \otimes \mathbf{1}_2) \cdot \mathrm{diag}(1, 1, 1, j) \cdot (\mathbf{1}_2 \otimes \mathrm{DFT}_2) \cdot [(2, 3), 4].$$

ACKNOWLEDGMENT

Part of the presented research was developed when the authors were 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, which made this work possible. The authors also want to thank Prof. J. M. F. Moura for helpful suggestions and proofreading.

REFERENCES

[1] Z. Wang and B. R. Hunt, "The discrete W Transform," *Appl. Math. Comput.*, vol. 16, pp. 19–48, 1985.
[2] M. T. Heideman, D. H. Johnson, and C. S. Burrus, "Gauss and the history of the fast Fourier transform," *Arch. History Exact Sci.*, vol. 34, pp. 265–277, 1985.
[3] J. W. Cooley and J. W. Tukey, "An algorithm for the machine calculation of complex Fourier series," *Math. Comput.*, vol. 19, pp. 297–301, 1965.
[4] C. M. Rader, "Discrete Fourier transforms when the number of data samples is prime," *Proc. IEEE*, vol. 56, pp. 1107–1108, 1968.
[5] S. Winograd, *Arithmetic Complexity of Computation*. Philadelphia, PA: SIAM, 1980.

[6] R. C. Agarwal and J. W. Cooley, "New algorithms for digital convolution," *IEEE Trans. Acoustics, Speech, Signal Processing*, vol. ASSP-25, pp. 392–410, 1977.

[7] L. I. Bluestein, "A linear filtering approach to the computation of the discrete Fourier transform," *IEEE Trans. Audio Electroacoust.*, vol. AU-18, pp. 451–455, Apr. 1970.

[8] M. Vetterli and H. J. Nussbaumer, "Simple FFT and DCT algorithms with reduced number of operations," *Signal Process.*, vol. 6, pp. 267–278, 1984.

[9] H. J. Nussbaumer, *Fast Fourier Transformation and Convolution Algorithms*, 2nd ed. New York: Springer, 1982.

[10] R. Tolimieri, M. An, and C. Lu, *Algorithms for Discrete Fourier Transforms and Convolution*, 2nd ed. New York: Springer, 1997.

[11] W.-H. Chen, C. H. Smith, and S. C. Fralick, "A fast computational algorithm for the discrete cosine transform," *IEEE Trans. Commun.*, vol. COMM-25, pp. 1004–1009, Sept. 1977.

[12] Z. Wang, "Fast algorithms for the discrete W transform and for the discrete Fourier transform," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-32, pp. 803–816, Apr. 1984.

[13] P. Yip and K. R. Rao, "Fast decimation-in-time algorithms for a family of discrete sine and cosine transforms," *Circuits, Syst., Signal Process.*, vol. 3, no. 4, pp. 387–408, 1984.

[14] ——, "The decimation-in-frequency algorithms for a family of discrete sine and cosine transforms," *Circuits, Syst., Signal Processing*, vol. 7, no. 1, pp. 3–19, 1988.

[15] B. Lee, "A new algorithm to compute the discrete cosine transform," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-32, pp. 1243–1245, June 1984.

[16] E. Feig, "A fast scaled-DCT algorithm," *Proc. SPIE*, vol. 1244, pp. 2–13, 1990.

[17] S. C. Chan and K. L. Ho, "Direct methods for computing discrete sinusoidal transforms," *Proc. Inst. Elect. Eng.*, vol. 137, no. 6, pp. 433–442, 1990.

[18] G. Steidl and M. Tasche, "A polynomial approach to fast algorithms for discrete Fourier-cosine and Fourier-sine transforms," *Math. Comput.*, vol. 56, no. 193, pp. 281–296, 1991.

[19] E. Feig and S. Winograd, "Fast algorithms for the discrete cosine transform," *IEEE Trans. Signal Processing*, vol. 40, pp. 2174–2193, Sept. 1992.

[20] L. Auslander, E. Feig, and S. Winograd, "Abelian semi-simple algebras and algorithms for the discrete Fourier transform," *Adv. Appl. Math.*, vol. 5, pp. 31–55, 1984.

[21] T. Beth, *Verfahren der Schnellen Fourier transformation*. Berlin, Germany: Teubner, 1984.

[22] D. Eberly and D. Wenzel, "Adaptation of group algebras to signal and image processing," *CVGIP: Graph. Models Image Process.*, vol. 53, no. 4, pp. 340–348, 1991.

[23] R. J. Valenza, "A representation-theoretic approach to the DFT with noncommutative generalizations," *IEEE Trans. Signal Processing*, vol. 40, pp. 814–822, Apr. 1992.

[24] D. Rockmore, "Some applications of generalized FFT's," in *Proc. DIMACS Workshop Groups Comput.*, vol. 28, 1995, pp. 329–370.

[25] A. Terras, *Fourier Analysis on Finite Groups and Applications*. Cambridge, U.K.: Cambridge Univ. Press, 1999.

[26] R. Foote, G. Mirchandi, D. Rockmore, D. Healy, and T. Olson, "A wreath product approach to signal and image processing: Part I—Multiresolution analysis," *IEEE Trans. Signal Processing*, vol. 48, pp. 102–132, Jan. 2000.

[27] T. Minkwitz, "Algorithmensynthese für lineare Systeme mit Symmetrie," Ph.D. dissertation, Univ. Karlsruhe, Dept. Informatik, Karlsruhe, Germany, 1993.

[28] ——, "Algorithms explained by symmetry," *Lecture Notes Comput. Sci.*, vol. 900, pp. 157–167, 1995.

[29] S. Egner, "Zur algorithmischen Zerlegungstheorie linearer Transformationen mit Symmetrie," Ph.D. dissertation, Univ. Karlsruhe, Dept. Informatik, Karlsruhe, Germany, 1997.

[30] M. Püschel, "Konstruktive Darstellungstheorie und Algorithmengenerierung," Ph.D. dissertation (in German), Univ. Karlsruhe, Dept. Informatik, Karlsruhe, Germany, 1998.

[31] ——, "Constructive representation theory and fast signal transforms," Drexel Univ., Philadelphia, PA, Tech. Rep. Drexel-MCS-1999-1 (translation of [30]), 1999.

[32] S. Egner and M. Püschel, "Fast discrete signal transforms and monomial representations of finite groups", submitted for publication.

[33] M. Püschel, "Decomposing monomial representations of solvable groups," Drexel Univ., Philadelphia, PA, Tech. Rep. Drexel-MCS-1999-2, 1999, submitted for publication.

[34] G. James and M. Liebeck, *Representations and Characters of Groups*. Cambridge, U.K.: Cambridge Univ. Press, 1993.

[35] M. R. Garey and D. S. Johnson, *Computers and Intractability*. New York: W. H. Freeman, 1979.

[36] S. Egner, M. Püschel, and GAP share package. (1998) AREP—Constructive representation theory and fast signal transforms. [Online]. Available: http://avalon.ira.uka.de/home/pueschel/arep/arep.html

[37] The GAP Team. (1997) GAP—Groups, algorithms, and programming. Univ. St. Andrews, St. Andrews, U.K.. [Online]. Available: http://www-gap.dcs.st-and.ac.uk/∼gap/

[38] K. R. Rao and P. Yip, *Discrete Cosine Transform: Algorithms, Advantages, Applications*. New York: Academic, 1990.

[39] H. S. Hou, "A fast recursive algorithm for computing the discrete cosine transform," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-35, pp. 1455–1461, Oct. 1987.

[40] L. Loeffler, "Practical fast 1-D DCT algorithms with 11 multiplications," in *Proc. ICASSP*, 1989, pp. 988–991.

[41] H. V. Sorensen, D. L. Jones, C. S. Burrus, and M. T. Heideman, "On computing the discrete hartley transform," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-33, pp. 1231–1238, Apr. 1985.

[42] D. F. Elliott and K. R. Rao, *Fast Transforms: Algorithms, Analyzes, Applications*: Academic Press, 1982.

[43] J. M. F. Moura, J. Johnson, R. W. Johnson, D. Padua, V. Prasanna, M. Püschel, and M. M. Veloso. (1998) SPIRAL: Portable Library of Optimized Signal Processing Algorithms. [Online]. Available: http://www.ece.cmu.edu/∼spiral/

[44] J. S. Leon, "Permutation group algorithms based on partitions, I: Theory and algorithms," *J. Symbolic Comput.*, vol. 12, no. 4/5, pp. 533–583, Oct./Nov. 1991.

Sebastian Egner was born in Hamburg, Germany, on April 5, 1968. He received the B.S. degree in computer science in 1989, the B.S. degree in physics in 1991, the M.S. degree in physics in 1994, and the Ph.D. degree in computer science in 1997, all from the University of Karlsruhe, Karlsruhe, Germany.

Since 1998, he has been with the Philips Research Laboratories, Eindhoven, The Netherlands, as a research scientist for signal processing and concepts of multimedia systems. He is interested in applied mathematics and computer science for problems in signal processing, error correcting codes, cryptography, and combinatorial optimization in multimedia systems and networks. Moreover, he has worked on algorithmic problems in computer algebra.

Markus Püschel received the M.Sc. degree in mathematics in 1995 and the Ph.D. degree in computer science in 1998, both from the University of Karlsruhe, Karlsruhe, Germany.

From 1998 to 1999, he was a Post-Doctoral Associate with the Department of Mathematics and Computer Science, Drexel University, Philadelphia, PA. Since 2000, he has held a Research Faculty position with the Department of Electrical and Computer Engineering, Carnegie Mellon University, Pittsburgh, PA. His research interests include signal and image processing, high-performance computing, algorithm theory, computer algebra, and quantum computing.