Fast Minimization of Polynomial Decomposition using Fixed-Polarity Pascal Transforms

Kaitlin N. Smith[1] and Mitchell A. Thornton
Department of Electrical and Computer Engineering
Southern Methodist University
Dallas, Texas, U.S.A.
{knsmith,mitch}@smu.edu

D. Michael Miller
Department of Computer Science
University of Victoria
Victoria, British Columbia, Canada
mmiller@uvic.ca

***Abstract*—Polynomials can be represented as a weighted sum of various powers of binomials where the weights are spectral coefficients and the associated powers of binomials are basis functions. The minimization problem is concerned with finding a set of basis functions that result in a maximum number of zero-valued spectral coefficients, or alternatively, wherein the spectral vector has a norm that is as close to zero as possible. One application of this minimization problem includes compression of signals that are represented with fitted polynomials. This problem can be considered to be a higher-radix generalization of the fixed-polarity Reed-Muller minimization problem since polynomial decomposition can be efficiently accomplished using the properties of the fixed-polarity family of Pascal transforms. We devise an efficient decomposition technique based on properties of the Pascal transform and we formulate a heuristic minimization algorithm to search for the minimal decomposition. Experimental results are provided that compare the quality and performance of the heuristic search to an exhaustive search for the minimal decomposition. The experimental results indicate that finding such a minimal decomposition can be achieved in a practical amount of runtime.**

I. Introduction

The Pascal transform can be represented by a square matrix with component values that are either zero or a binomial coefficient denoted as $\binom{n}{k}$ and referred to as "n choose k." In general, the Pascal transform is a matrix of infinite dimension since the integer n increases without bound. However, practical uses of the transform limit n to some upper value resulting in a truncated, finite-dimensioned $(n+1)\times(n+1)$ square transformation matrix. We shall refer to the truncated Pascal transform as the "Pascal transform."

The Pascal transform has several applications. In terms of filter theory, the transform has found uses that include continuous s- to discrete z-domain conversion [1], various image processing filters [2], transformation of analog lowpass to digital bandpass filters [3], and in defining a new class of filters entirely [4]. With regard to topics in mathematics, there are many applications and relationships based on the transform including those described in [5]–[10]. With respect to computational algorithms, the recursive definition of the Pascal transform matrix as well as its orthogonality have enabled the development of efficient or "fast" algorithms [11]–[14]. Pascal transforms are also of interest in the binary and MVL switching theory communities for a variety of reasons including the selected examples disclosed in [10], [15].

The set of k^{th}-degree polynomials $p(x)$ that result from the expansion and simplification of the k^{th} power of the binomial of the form $(x+1)^k$ are explicitly represented as row or column vectors in the Pascal transform matrix for all $k \leq n$ since the components in the k^{th} row or column vector are the coefficients of $p(x)$. It should be noted that the index, k, in reference to the row and column vectors of the transformation matrices discussed in this paper have values $0 \leq k \leq n$.

The generalized family of Pascal transforms disclosed in [16] is based on concepts in binary-valued switching theory due to the observation that the radix-2 Zhegalkin polynomial is related to the positive-polarity Reed-Muller (PPRM) transformation matrix in the same way that polynomials of the form $p(x) = (x+1)^k$ are related to the Pascal transform [17]. In fact, the PPRM transformation matrix can be computed by replacing the Pascal matrix components with their modulo-2 value. Thus, the concept of a polarity number can be used to determine whether each row or column vector comprising the Pascal matrix is related to the $(x+1)^k$ or the $(x-1)^k$ term. In this manner, the polarity number is formed as a binary value where "1" represents a $(x-1)^k$ term and "0" represents a $(x+1)^k$ term. Thus, there exist a total of 2^n different matrices based upon the polarity value.

We are interested in a family of Pascal transform matrices that can be further extended from the form described in [16]. In particular, we are interested in the family of transforms where the form of the binomial associated with each row or column vector is generalized and of the form, $(x \pm d_k)^k$, where d_k is any value, $d_k \in \mathbb{Z}$. In this case, rather than using a polarity number to characterize the Pascal transform matrix, we use a polarity vector, $\mathbf{d}$, of dimension n with associated components, d_k.

A family of matrices comprise the inverses of generalized Pascal matrices associated with each polarity vector $\mathbf{d}$. They are of interest as they are used to find the representation of an n^{th}-degree polynomial, $p(x)$, as a weighted sum of binomials that are raised to the power k where $0 \leq k \leq n$. We refer to this form of representation of $p(x)$ as a "polynomial decomposition." The computation of this decomposition form can be accomplished by assuming a particular polarity vector,

[1]K. N. Smith was with Southern Methodist University. She is now at the Department of Computer Science at the University of Chicago (email: kns@uchicago.edu).

$\mathbf{d}$, formulating the associated Pascal transformation matrix of the size $(n+1) \times (n+1)$ where n is the degree of the polynomial to be decomposed, computing the inverse of the transformation matrix, and finally, computing the direct vector matrix product of the inverse matrix with the polynomial coefficient vector, $\mathbf{a}$. The resultant product vector $\mathbf{b}$ then yields the multiplicative scalar weights associated with each binomial term, $(x+d_k)^k$. In this manner, the polynomial $p(x)$ is represented as a sum of weighted binomials in the form of $b_k(x+d_k)^k$ for $k=0$ to $k=n$. The sum of the weighted binomials is then the binomial decomposition of the polynomial, $p(x)=\sum_{i=0}^{n} b_i(x+d_i)^i$.

The objective of this paper is to find the polarity vector, $\mathbf{d}$, such that the resulting vector $\mathbf{b}$ contains as many zero-valued coefficients as possible. In the event that there is no $\mathbf{b}$ vector that contains at least one zero-valued component, then we wish to find $\mathbf{d}$ that minimizes $\mathbf{b}$. We define the minimum weight $\mathbf{b}$ vector to be the vector whose L_1 or L_2 norms in L_p, or Lebesgue, spaces is minimized. We refer to the process of finding the polarity vector that results in a minimum weight $\mathbf{b}$ vector as the "minimization of polynomial decomposition" problem. When $b_i = 0$, it is not necessary to know the explicit value of the corresponding d_i to recreate $p(x)$ since the d_i value is multiplied by its corresponding zero-valued b_i coefficient.

The problem of finding the minimal polynomial decomposition can be considered an optimization problem for $p(x)$ when it is represented as a weighted sum of powers of binomials. We consider the decomposition to be optimal in the sense that it maximizes the number of zero-valued components in the $\mathbf{b}$ vector. From this point of view, finding the polarity vector that maximizes the compression of the polynomial is analogous to the problem of representing a space- or time-varying signal with as few non-zero spectral coefficients as possible. This is a frequently encountered task in the efficient representation of signals and images for efficient transmission. Alternatively, the solution of this problem can also be viewed as a higher-radix generalization of the problem of finding the minimal polarity number for a fixed-polarity Reed-Muller representation of a binary switching function. This causes the problem to be of interest to the multiple-valued logic community. The representation of a polynomial as a weighted sum of powers of particular binomials is mathematically a type of spectral representation wherein the various binomials raised to integer powers comprise a set of basis functions that define the particular transformation matrix for the decomposition. In solving the minimization problem, we seek the basis function set that yields the most compact spectral representation of the polynomial. Furthermore, the identification of the appropriate polarity vector, $\mathbf{d}$, is equivalent to finding this optimal basis set.

II. Background and Theoretical Concepts

A. Pascal's Triangle

Pascal's triangle is an explicit arrangement of the combinatorial coefficients in Eqn. 1 where the nth row of the triangle contains the coefficients beginning with coefficient of the nth-degree monomial on the left to the 0th-degree monomial (or constant) on the right, or vice-versa.

$$(x+y)^n = \sum_{i=0}^{n} \binom{n}{i} x^i y^{n-i}. \qquad (1)$$

We refer to the values in each row, the binomial coefficients, as a_k where $k=0,\cdots,n$. These values are also the coefficients of the simplified polynomial form of $(x+1)^n$. We use the term "simplified polynomial" to denote that the polynomial is comprised of only a single term for each degree value n. That is, each instance of the monomial, $a_k x^k$, for a specific k appears only once in the simplified and expanded polynomial form of $(x+1)^n$ as shown in Table I.

TABLE I
Pascal's Triangle with Corresponding Binomials and Simplified Polynomials

Binomial Power	Binomial	Polynomial
$n=0$	$(x+1)^0$	$(1)x^0$
$n=1$	$(x+1)^1$	$(1)x^1+(1)x^0$
$n=2$	$(x+1)^2$	$(1)x^2+(2)x^1+(1)x^0$
$n=3$	$(x+1)^3$	$(1)x^3+(3)x^2+(3)x^1+(1)x^0$
$n=4$	$(x+1)^4$	$(1)x^4+(4)x^3+(6)x^2+(4)x^1+(1)x^0$

A modified form of Pascal's triangle exists that corresponds to binomials of the form $(x-1)^n$. This well-known alternative form of Pascal's triangle is identical to that given previously with the exception that each alternating value in the triangle is negated. The modified Pascal's triangle is found in Table II.

TABLE II
Modified Pascal's Triangle in Terms of Simplified Binomials

Binomial Power	Binomial	Polynomial
$n=0$	$(x-1)^0$	$(1)x^0$
$n=1$	$(x-1)^1$	$(1)x^1-(1)x^0$
$n=2$	$(x-1)^2$	$(1)x^2-(2)x^1+(1)x^0$
$n=3$	$(x-1)^3$	$(1)x^3-(3)x^2+(3)x^1-(1)x^0$
$n=4$	$(x-1)^4$	$(1)x^4-(4)x^3+(6)x^2-(4)x^1+(1)x^0$

B. The Pascal Transform

The Pascal transform refers to a linear transformation matrix that contains components based upon the Pascal triangle [5], [7]. The Pascal's transform matrix can be expressed in a lower triangular, $\mathbf{P}_L$, or a upper triangular, $\mathbf{P}_U$, form. Pascal's matrices are infinite in dimension, however, it is convenient and common to utilize truncated versions of finite dimension based upon the degree of the polynomial of interest. The rows and columns of the Pascal matrices have an index encoding that begins at zero so that the index values are equivalent to the binomial exponent in the triangle. In truncated form, the two different forms of the Pascal transformation matrices are denoted as a finite dimensioned $k \times k$ form as $\mathbf{P}_{Lk}$ and $\mathbf{P}_{Uk}$ where the indices are $i,j=0,1,\ldots,k-1$, and $k-1=n$. Eqn. 2 contains the 4×4 truncated versions of the Pascal transformation matrices.

$$\mathbf{P}_{L4} = \begin{bmatrix} 1&0&0&0 \\ 1&1&0&0 \\ 1&2&1&0 \\ 1&3&3&1 \end{bmatrix} \quad \mathbf{P}_{U4} = \begin{bmatrix} 1&1&1&1 \\ 0&1&2&3 \\ 0&0&1&3 \\ 0&0&0&1 \end{bmatrix} \tag{2}$$

The construction definitions for the lower and upper triangular forms based on the binomial coefficients are provided in Eqn. 3 and Eqn. 4.

$$\begin{aligned} &\mathbf{P}_L = [p_{L;i,j}] \\ &p_{L;i,i} = p_{L;i,0} = 1 \forall i = 0, \dots, n-1, \quad p_{L;i,j} = 0 \text{ if } j > i \\ &p_{L;i,j} = \begin{pmatrix} i \\ j \end{pmatrix} \quad \forall i,j = 0, \dots, n-1 \text{ and } i > j \end{aligned} \tag{3}$$

$$\begin{aligned} &\mathbf{P}_U = [p_{U;i,j}] \\ &p_{U;i,i} = p_{U;0,i} = 1 \forall i = 0, \dots, n-1, \quad p_{U;i,j} = 0 \text{ if } j < i \\ &p_{U;i,j} = \begin{pmatrix} j \\ i \end{pmatrix} \quad \forall i,j = 0, \dots, n-1 \text{ and } i < j \end{aligned} \tag{4}$$

With respect to the binomial theorem, modified Pascal matrices can be formed with respect to the modified Pascal triangle that contain rows corresponding to simplified polynomial forms of $(x-1)^n$. These are provided in Eqn. 5.

$$\mathbf{P}_{ML4} = \begin{bmatrix} 1&0&0&0 \\ -1&1&0&0 \\ 1&-2&1&0 \\ -1&3&-3&1 \end{bmatrix} \quad \mathbf{P}_{MU4} = \begin{bmatrix} 1&-1&1&-1 \\ 0&1&-2&3 \\ 0&0&1&-3 \\ 0&0&0&1 \end{bmatrix} \tag{5}$$

The modified Pascal matrices may also be defined using the binomial coefficients.

$$\begin{aligned} &\mathbf{P}_{ML} = [p_{ML;i,j}] \\ &p_{ML;i,i} = p_{ML;i,0} = (-1)^{i+j} \forall i = 0, \dots, n-1, p_{ML;i,j} = 0 \text{ if } i < j \\ &p_{ML;i,j} = (-1)^{i+j} \begin{pmatrix} i \\ j \end{pmatrix} \forall i,j = 1, \dots, n-1 \text{ and } i > j \end{aligned} \tag{6}$$

$$\begin{aligned} &\mathbf{P}_{MU} = [p_{MU;i,j}] \\ &p_{MU;i,i} = p_{MU;0,i} = (-1)^{i+j} \forall i = 0, \dots, n-1, p_{MU;i,j} = 0 \text{ if } i > j \\ &p_{MU;i,j} = (-1)^{i+j} \begin{pmatrix} j \\ i \end{pmatrix} \forall i,j = 1, \dots, n-1 \text{ and } i < j \end{aligned} \tag{7}$$

Note that the forms of the Pascal and modified Pascal transformation matrices are related by $\mathbf{P}_{Lk} = \mathbf{P}_{Uk}^T$ and $\mathbf{P}_{MLk} = \mathbf{P}_{MUk}^T$, respectively.

The various Pascal and modified Pascal matrices have properties that allow for the transformation of polynomial functions. For instance, the Pascal transformation matrices $\mathbf{P}_L = [p_L; i,j]$ and $\mathbf{P}_U = [p_U; i,j]$ are "unitriangular" meaning they are triangular matrices with diagonal elements $p_L; i,i = p_U; i,i = 1$. Since the determinant is equal to the product of the diagonal elements, and all of the diagonal elements are unity, the determinants of $\mathbf{P}_L$, $\mathbf{P}_U$, $\mathbf{P}_{ML}$, and $\mathbf{P}_{MU}$, are equal to one. Another important characteristic of the Pascal and modified Pascal matrices is their relationship through inversion such that

$$\mathbf{P}_U^{-1} = \mathbf{P}_{MU} \tag{8}$$

and

$$\mathbf{P}_{MU}^{-1} = \mathbf{P}_U. \tag{9}$$

C. Pascal's Matrix for Polynomial Simplification

One of the most well-known applications of Pascal's triangle is its use in representing simplified polynomials that correspond to a binomial of the form $(x+1)$ raised to some positive integer power n as shown in Table I or to a binomial of the form $(x-1)$ raised to some positive integer power n as shown in Table II. To convert from binomial to simplified polynomial form,

$$\mathbf{a} = \mathbf{P}\mathbf{b} \tag{10}$$

is evaluated where $\mathbf{b}$ holds the binomial coefficients, $\mathbf{a}$ holds the polynomial coefficients, and $\mathbf{P}$ is either $\mathbf{P}_U$ or $\mathbf{P}_{MU}$.

D. Fixed Polarity Pascal Transforms

The preceding section described how the simplified polynomial expansion of linear combinations of binomials of the form $(x+1)^n$ and $(x-1)^n$ can be obtained using Pascal's matrix. Polynomial expressions with mixed forms of binomials can be formulated.

A family of the upper triangular version of Pascal's transformation matrix may be formulated by selecting some column vectors from $\mathbf{P}_U$ and the remaining column vectors from $\mathbf{P}_{MU}$. The resulting matrix, denoted $\mathbf{P}_{pU}$ is characterized by an integer p referred to as the "polarity number" of this form of Pascal's matrix. The polarity number is the decimal equivalent of the binary string formed by assigning a zero (0) to those columns from $\mathbf{P}_U$ and a one (1) to those columns from $\mathbf{P}_{MU}$. The string is read from left to right with the most significant bit corresponding to the binomial exponent $n = 0$. We refer to these forms of Pascal's transformation matrices as the fixed-polarity form since each polarity number refers to a specific matrix structure.

The "fixed-polarity Pascal transform" (FPPT) matrices can be applied in determining the simplified polynomial expansions of weighted linear combinations of powers of binomials of the form $(x+1)^n$ and $(x-1)^n$. While this method offers a convenient method and one that potentially requires less effort than manual symbolic polynomial multiplication, the inverse problem of finding a decomposition of a polynomial into a weighted sum of binomials is more difficult and is the focus of this paper. This calculation is of importance since we are concerned with finding the coefficients, b_i, associated with the binomial expansion of a polynomial, $p(x)$, that is characterized by its corresponding $\mathbf{a}$ vector expressed as

$$\mathbf{b} = \mathbf{P}^{-1}\mathbf{a} \tag{11}$$

.

The notation for a FPPT matrix with a polarity p is defined as $\mathbf{P}_{pUk}$. The polarity value p indicates whether the i^{th} column vector of $\mathbf{P}_{pUk}$ is equivalent to that of $\mathbf{P}_{Uk}$ or $\mathbf{P}_{MUk}$ if it has a value of 0 or 1, respectively, in its binary form. The dimension of the matrix is indicated by k and is related to the largest integer power n of the polynomial or binomial that can be processed by $\mathbf{P}_{pUk}$ matrix as $k = n+1$. Thus, the polarity number for the upper triangular matrix in Eqn. 2 would be $p = 0000_2 = 0_{10}$ to form $\mathbf{P}_{0U4}$ and the polarity number for the upper triangular matrix in Eqn. 5 would be $p = 1111_2 = 15_{10}$ to form $\mathbf{P}_{15U4}$. These transforms can be referred to as the positive-polarity and negative-polarity Pascal transforms, respectively. An example of a mixed polarity transform for $n = 3$ and $p = 0011_2 = 3_{10}$ is

$$\mathbf{P}_{3U4} = \begin{bmatrix} 1 & 1 & 1 & -1 \\ 0 & 1 & -2 & 3 \\ 0 & 0 & 1 & -3 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{12}$$

The positive-polarity, $\mathbf{P}_{0U4} = \mathbf{P}_{U4}$ and negative-polarity, $\mathbf{P}_{15U4} = \mathbf{P}_{MU4}$ Pascal transforms, as shown in Eqns. 8 and 9, are inverses of each other. This property simplifies the process of applying Eqn. 11 for $\mathbf{b}$ whenever p holds only zeros or ones. Determining the inverse of a mixed polarity Pascal transform is slightly more complex, but it can be accomplished with algorithms such as the those described in [16].

To further generalize the polynomial decomposition in terms of a polarity vector, $\mathbf{d}$, the inverse matrix, $\mathbf{P}^{-1}$, is based upon the use of a $\mathbf{P}$ matrix where each column vector corresponds to the coefficients of an expanded polynomial of the form $(x+d_k)^k$ where the vector $\mathbf{d}$ is comprised of components $d_i \in \mathbb{Z}$. It is noted that this relationship allows a polynomial, $p(x)$ to be expressed either as a polynomial coefficient vector $\mathbf{a}$, or alternatively, as a vector of "pairs," (b_i, d_i). This relationship is expressed in equation form as

$$p(x) = \sum_{i=0}^{n} a_i x^i = \sum_{i=0}^{n} b_i (x + d_i)^i \tag{13}$$

.

The inverse Pascal transform matrices that are formulated with a polarity vector fortunately have constructive definitions and can be generated explicitly without resorting to formulating the forward Pascal transformation matrix and then using a general algorithm to find the inverse matrix. Additionally, due to the triangular property of the inverse matrix, the forward matrix can be used with an iterative back-substitution algorithm to solve for the $\mathbf{b}$ vector thus avoiding the need to compute the inverse matrix entirely. Both of these results are derived and shown in [16].

III. Binomial Decomposition Minimization

The FPPT can be used to transform a polynomial into a sum of weighted binomials in the form of $(x+1)^n$ and $(x-1)^n$. In a polynomial of degree n, there are a total of 2^n polarity values p that can be used to form the transformation matrix associated with the FPPT. Because each polarity creates a different $\mathbf{P}_{pUk}$, and thus a corresponding $\mathbf{P}_{pUk}^{-1}$, the final weights of $\mathbf{b}$ are highly influenced by p. For example, consider finding the binomial coefficients for the polynomial $1+3x+3x^2+x^3$ with the coefficient vector of $\mathbf{a}^T = [1, 3, 3, 1]$ using $p = 0000_2$ to form the transform $\mathbf{P}_{0U4}$. Knowing that $\mathbf{P}_{0U4}^{-1} = \mathbf{P}_{15U4}$, $\mathbf{b}$ is calculated as

$$\mathbf{b} = \mathbf{P}_{15U4}\mathbf{a} = \begin{bmatrix} 1 & -1 & 1 & -1 \\ 0 & 1 & -2 & 3 \\ 0 & 0 & 1 & -3 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 3 \\ 3 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}.$$

Thus, $1 + 3x + 3x^2 + x^3 = (x + 1)^3$. The L_1 distance norm of $\mathbf{b}$ for $p = 0000_2$ is 1 and the L_2 distance norm of $\mathbf{b}$ for $p = 0000_2$ is 1. If the FPPT from Eqn. 12 with polarity $p = 0011_2$ was used to find $\mathbf{b}$ from $\mathbf{a}^T = [1, 3, 3, 1]$, a different binomial decomposition would result. This is shown by

$$\mathbf{b} = (\mathbf{P}_{3U4})^{-1}\mathbf{a} = \begin{bmatrix} 1 & -1 & -3 & -5 \\ 0 & 1 & 2 & 3 \\ 0 & 0 & 1 & 3 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 3 \\ 3 \\ 1 \end{bmatrix} = \begin{bmatrix} -16 \\ 12 \\ 6 \\ 1 \end{bmatrix}$$

resulting in the binomial $-16(x + 1)^0 + 12(x + 1) + 6(x - 1)^2 + (x - 1)^3 = 1 + 3x + 3x^2 + x^3 = (x + 1)^3$. The L_1 distance norm of $\mathbf{b}$ for $p = 0011_2$ is 35 and the L2 norm of $\mathbf{b}$ for $p = 0011_2$ is approximately 12.905. Since the distance of $p = 0011_2$ is greater than that of $p = 0000_2$, $p = 0000_2$ can be considered a more minimized FPPT for the polynomial $1+3x+3x^2+x^3 = (x+1)^3$. The calculated L_1 and L_2 distance norms (rounded to three decimals), total zero coefficients, and resulting $\mathbf{b}$ vectors for all polarities can be found in Table III.

TABLE III
Calculated distance of binomial coefficients, $\mathbf{b}$, for $\mathbf{a}^T = [1, 3, 3, 1]$ at various FPPT polarities, p

p	$\mathbf{b}^T$	# zero coeff.	L_1 **distance**	L_2 **distance**
0000_2	$[0, 0, 0, 1]$	3	1	1
0001_2	$[8, -12, 6, 1]$	0	27	15.652
0010_2	$[0, 0, 0, 1]$	3	1	1
0011_2	$[-16, 12, 6, 1]$	0	35	20.905
0100_2	$[0, 0, 0, 1]$	3	1	1
0101_2	$[-16, 12, 6, 1]$	0	35	20.905
0110_2	$[0, 0, 0, 1]$	3	1	1
0111_2	$[8, 12, 6, 1]$	0	27	15.652
1000_2	$[0, 0, 0, 1]$	3	1	1
1001_2	$[8, -12, 6, 1]$	0	27	15.652
1010_2	$[0, 0, 0, 1]$	3	1	1
1011_2	$[-16, 12, 6, 1]$	0	35	20.905
1100_2	$[0, 0, 0, 1]$	3	1	1
1101_2	$[-16, -12, 6, 1]$	0	35	20.905
1110_2	$[0, 0, 0, 1]$	3	1	1
1111_2	$[8, 12, 6, 1]$	0	27	15.652

As observed in Table III, polarity values that allowed for a larger number of zero coefficients were most favorable in terms of minimized binomial representation. If zero coefficients are not present in $\mathbf{b}$, then distance of $\mathbf{b}$ should be calculated via L_1 and L_2 distance norms in order to determine the polarity value that allows for the smallest weights in $\mathbf{b}$. Another important observation from Table III is that weights can appear at different polarities that have equal distances. An example of this repetition lies in the fact that both halves of the table representing the ranges of $p = [0000_2 : 0111_2]$

and $p = [1000_2 : 1111_2]$ are identical. This repetition appears because the polarity of the first term in the binomial expression is irrelevant since it is raised to the zeroth power (i.e. $(x+1)^0 = (x-1)^0$).

As another example, consider the family of Laguerre polynomials for degree $n = [2:6]$:

$$\frac{1}{2}(x^2 - 4x + 2) \tag{14}$$

$$\frac{1}{6}(-x^3 + 9x^2 - 18x + 6) \tag{15}$$

$$\frac{1}{24}(x^4 - 16x^3 + 72x^2 - 96x + 24) \tag{16}$$

$$\frac{1}{120}(-x^5 + 25x^4 - 200x^3 + 600x^2 - 600x + 120) \tag{17}$$

$$\frac{1}{720}(x^6 - 36x^5 + 450x^4 - 2400x^3 + 5400x^2 - 4320x + 720) \tag{18}$$

For Eqns. 14-18, the minimized polarity for a binomial decomposition can be found with the FPPT according to the minimizing criteria previously described:

1) Search through polarities for one that calculates a $\mathbf{b}$ with a maximum number of zero coefficients.
2) If no set of weights, $\mathbf{b}$, contains zero, determine minimum L_1 distance for all polarities and return all polarity values equal to the minimum distance.
3) If no set of weights, $\mathbf{b}$, contains zero, determine minimum L_2 distance for all polarities and return all polarity values equal to the minimum distance.

as a note, the fractional scalar was omitted from $\mathbf{a}$ to keep the polynomial weights integer values. No polarity produced zero coefficients in $\mathbf{b}$, so the calculated distance was used to find the minimal decomposition. The results for polarization minimization of the Laguerre polynomials based off distance calculations can be found in Table IV.

IV. Branch and Bound Method

The previous results performed an exhaustive search over all polarity numbers to find the minimum. Clearly, this is an impractical approach for polynomials of large degree. Thus, a heuristic approach based upon a branch and bound paradigm is formulated as follows. Given a polynomial $p(x) = \sum_{i=0}^{n} a_i x^i$, the procedure FACTOR performs a branch and bound recursive search to find a binomial factorization of that polynomial $\sum_{i=0}^{n} b_i(x+d_i)^i$, all $d_i \in \{+1, -1\}$ for $i > 0$, with a minimum number of $b_i \neq 0$. The initial call to FACTOR should be FACTOR(a, b, d, n) where a specifies the n-degree polynomial to be factored and b and d have all entries set to 0. The final binomial factorization is placed in global values $BESTb$ and $BESTd$ that are initially undefined. The function $nterm$ returns the number of nonzero entries in its argument. It returns $n+1$ if the argument is not yet defined.

Lines 6-11 handle the terminal case for the recursion at which point the current expansion (in b and d) is checked to see if it has fewer terms than the best solution found so far. Lines 13-15 check the current expansion and prune the search if it can not lead to a solution better than the current best.

```
1: procedure FACTOR($a, b, d, n$)
2:     $k \leftarrow n$
3:     while $k > 0$ and $a_k = 0$ do
4:         $k \leftarrow k - 1$
5:     end while
6:     if $k = 0$ then
7:         $b_0 \leftarrow a_0$, $d_0 \leftarrow 0$
8:         if $nterm(b) < nterm(BESTb)$ then
9:             $BESTb \leftarrow b$
10:            $BESTd \leftarrow d$
11:        end if
12:    else
13:        if $nterm(b) = nterm(BESTb) - 1$ then
14:            return
15:        end if
16:        for $u = 1, -1$ do
17:            $b_k \leftarrow a_k$
18:            $q_k \leftarrow u$
19:            $a1 \leftarrow a$ minus polynomial expansion of $a_k(x+u)^k$
20:            FACTOR($a1, b, q, k-1$)
21:        end for
22:    end if
23:    return
24: end procedure
```

To clarify the approach, we first consider using only terms of the form $v(x+1)^k$. For example, consider the polynomial $2x^3 + 7x^2 + 4x + 3$. To begin the binomial expansion has no terms. The highest order term of the polynomial is $2x^3$ so we add $2(x+1)^3$ to the expansion and subtract $2(x^3 + 3x^2 + 3x + 1)$ from the polynomial so we now have the polynomial $x^2 - 2x + 1$ and expansion $2(x+1)^3$. The highest order term of the polynomial is now x^2 so we add $(x+1)^2$ to the binomial expansion to get $2(x+1)^3 + (x+1)^2$ and subtract $x^2 + 2x + 1$ from the polynomial leaving $-4x$.

The next step is to add $-4(x+1)$ to the expansion giving $2(x+1)^3 + (x+1)^2 - 4(x+1)$ and subtract the same from the polynomial giving 4. The final step is to subtract 4 from the polynomial and add 4 to the expansion giving $2(x+1)^3 + (x+1)^2 - 4(x+1) + 4$ which is the final expansion since the polynomial is 0.

The above is an iterative approach that is the initial solution found by the recursive procedure FACTOR. By backtracking and trying terms of the form $v(x-1)^k$ in place of each choice of $v(x+1)^k$ the recursive search finds the best solution for this example to be $2(x+1)^3 + (x-1)^2$ which has two terms.

Note that the search is not exhaustive over all polarity assignments as it is pruned by ignoring search paths that will lead to a solution with the same or a higher number of terms than the best solution found so far during the search. More elaborate pruning rules could be used if just minimizing the number of terms is not sufficient.

Table V provides a timing comparison between the bounded search and brute force search methods.

V. Conclusion

We have defined a minimization problem for polynomial decomposition that has applications including lossless compression of signals represented by fitted polynomials. This problem

TABLE IV
POLARIZATION MINIMIZATION FOR LAGUERRE POLYNOMIALS

Polynomial #	Minimized L_1 p (# matching)	Minimized L_1 $\mathbf{b}$	Minimized L_2 p (# matching)	Minimized L_2 $\mathbf{b}^T$
2	011_2 (2)	$[-1, -2, 1]$	011_2 (2)	$[-1, -2, 1]$
3	0011_2 (2)	$[2, -3, 6, -1]$	0011_2 (2)	$[2, -3, 6, -1]$
4	01111_2 (2)	$[-15, 4, 30, -12, 1]$	01111_2 (2)	$[-15, 4, 30, -12, 1]$
5	011111_2 (2)	$[-56, 95, 140, -110, 20, -1]$	011111_2 (2)	$[-56, 95, 140, -110, 20, -1]$
6	0001110_2 (4)	$[-417, -546, 315, -1100, 225, -42, 1]$	0001110_2 (2)	$[-417, -546, 315, -1100, 225, -42, 1]$

TABLE V
TIMING COMPARISON (CPU SEC.) BETWEEN BRANCH AND BOUND AND BRUTE FORCE SEARCH WHEN CALCULATING $\mathbf{b}^T$

degree	$\mathbf{a}^T$	p	$\mathbf{b}^T$	**heuristic timing**	**brute force timing**
4	$[0, 7, 3, 5, 1]$	00010_2	$[0, 0, 0, 1, 1]$	2.32E-5	2.85E-2
5	$[-2, 8, -13, 11, -5, 1]$	000101_2	$[0, 0, 0, 1, 0, 1]$	2.94E-5	6.49E-2
6	$[2, 2, 21, 16, 16, 6, 1]$	0000100_2	$[0, 0, 0, 0, 1, 0, 1]$	2.64E-5	1.35E-1
7	$[2, 9, 22, 35, 35, 21, 7, 1]$	00000000_2	$[0, 0, 1, 0, 0, 0, 0, 1]$	1.74E-5	3.09E-1
8	$[0, 5, 25, 55, 70, 56, 28, 8, 1]$	000000000_2	$[0, 0, 0, -1, 0, 0, 0, 0, 1]$	2.44E-5	7.31E-1
9	$[4, 11, 55, 79, 132, 125, 84, 36, 9, 1]$	0000010000_2	$[0, 0, 0, 1, 1, -1, 0, 0, 0, 1]$	3E-5	1.71
10	$[2, 18, 64, 156, 245, 273, 217, 121, 45, 10, 1]$	00110000000_2	$[0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 1]$	2.52E-5	3.97
11	$[2, 8, 46, 149, 316, 456, 461, 330, 165, 55, 11, 1]$	010000000000_2	$[0, -1, 0, 0, 1, 0, -1, 0, 0, 0, 0, 1]$	2.4E-5	9.43
12	$[0, 4, 78, 182, 529, 771, 931, 791, 495, 220, 66, 12, 1]$	0001000100000_2	$[0, 0, 0, 1, -1, 0, 0, -1, 0, 0, 0, 0, 1]$	6.04E-5	22.7
13	$[0, 12, 62, 255, 681, 1266, 1709, 1715, 1287, 715, 286, 78, 13, 1]$	00100000000000_2	$[0, 0, -1, 0, 1, 0, 0, -1, 0, 0, 0, 0, 0, 1]$	2.6E-5	51.8

can also be considered as higher-radix generalization of the fixed-polarity Reed-Muller minimization problem in binary-valued switching theory. We have shown that the computation of a polynomial decomposition can be efficiently accomplished using previous results that exploit the structure of the fixed-polarity Pascal transformation matrices thus alleviating the need to explicitly compute the inverse transformation matrices. Furthermore, we have devised, implemented, and evaluated a heuristic searching method that finds the minimal decomposition given a finite set of polarity vectors. This implementation has practical runtime characteristics, allowing the method to be applicable for high-degree polynomials.

REFERENCES

[1] T.-B. Deng, S. Chivapreecha, and K. Dejhan, "Unified Pascal matrix for first-order s - z domain transformations," *IEEE Transactions on Signal Processing*, vol. 57, no. 6, pp. 3090–3094, 2009.

[2] M. F. Aburdene and T. J. Goodman, "The discrete Pascal transform and its applications," *IEEE Signal Processing Letters*, vol. 12, no. 7, pp. 493–495, 2005.

[3] Y. Choo and C. G. K., "On Pascal matrix for transforming lowpass analog to bandpass digital filter," *IEEE Signal Processing Letters*, vol. 16, no. 2, pp. 77–80, 2009.

[4] T. J. Goodman and M. F. Aburdene, "Pascal filters," *IEEE Transactions on Circuits and Systems–I*, vol. 55, no. 10, pp. 2130–2139, 2008.

[5] G. S. Call and D. J. Velleman, "Pascal's matrices," *The American Mathematical Monthly*, vol. 100, no. 4, pp. 372–376, 1993.

[6] M. F. Aburdene and J. E. Dorband, "Unification of Legendre, Laguerre, Hermite, and binomial discrete transforms using Pascal's matrix," *Multidimensional Systems and Signal Processing*, vol. 5, no. 3, pp. 301–305, 1994.

[7] A. Edelman and G. Strang, "Pascal matrices," http://web.mit.edu/18.06/www/ Essays/pascal-work.pdf, MIT, Cambridge, MA, Tech. Rep., 2003.

[8] S. M. Z. and L. L. Steil, "The k-binomial transforms and the Hankel transform," *Journal of Integer Sequences*, vol. 9, no. 1, pp. 06.1.1:1–19, 2006.

[9] T. J. Goodman and M. F. Aburdene, "On discrete Pascal transform, Poisson sequence and Laguerre polynomials," *Electronics Letters*, vol. 43, no. 14, 2007.

[10] C. Moraga, R. Stanković, and M. Stanković, "The Pascal triangle (1654), the Reed-Muller-Foutier transform (1992), and the discrete Pascal transform (2005)," in 46th *Int. Symp. on Multiple-valued Logic*. IEEE Computer Press, 2016, pp. 229–234.

[11] M. Aburdene, R. Kozick, R. Magargle, J. Maloney, and C. Coviello, "Discrete polynomial transform representation using binary matrices and flow diagrams," in *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, vol. 2, 02 2001, pp. 1141 – 1144 vol.2.

[12] A. N. Skodras, "Fast discrete Pascal transform," *Electronics Letters*, vol. 42, no. 23, 2006.

[13] Q. Zhong, A. K. Nandi, and M. F. Aburdene, "Efficient implementation of discrete Pascal transform using difference operators," *Electronics Letters*, vol. 43, no. 24, 2007.

[14] D. B. Gajić and R. S. Stanković, "Fast computation of the discrete Pascal transform," in 47th *Int. Symp. on Multiple-valued Logic*. IEEE Computer Press, 2017, pp. 149–154.

[15] R. S. Stanković, J. Astola, and C. Moraga, "Pascal matrices, Reed-Muller expressions and Reed-Muller error correcting codes," *Zbornik Radova*, vol. 18, no. 26, pp. 145–172, 2015.

[16] K. Smith and M. Thornton, "Fixed polarity Pascal transforms with symbolic computer algebra applications," in *Proceedings of IEEE Pacific Rim Conference on Communications, Computers, and Signal Processing (PacRim)*, 2019.

[17] V. Suprun and D. Gorodecky, "Matrix method of polynomial expansion of symmetric Boolean functions," *Automatic Control and Computer Sciences*, vol. 47, no. 1, pp. 1–6, 2013.