Brief Contributions

A Deterministic Multivariate Interpolation Algorithm for Small Finite Fields

Zeljko Zilic, *Member*, *IEEE*, and
Zvonko G. Vranesic, *Member*, *IEEE*

Abstract—We present a new multivariate interpolation algorithm over arbitrary fields which is primarily suited for small finite fields. Given function values at arbitrary t points, we show that it is possible to find an *n*-variable interpolating polynomial with at most t terms, using the number of field operations that is polynomial in t and n. The algorithm exploits the structure of the multivariate generalized Vandermonde matrix associated with the problem. Relative to the univariate interpolation, only the minimal degree selection of terms cannot be guaranteed and several term selection heuristics are investigated toward obtaining low-degree polynomials. The algorithms were applied to obtain Reed-Muller and related transforms for incompletely specified functions.

Index Terms—Multivariate interpolation, finite fields, Vandermonde matrices, Reed-Muller transform.

1 Introduction and Background

THIS paper presents an algorithm for multivariate polynomial interpolation, developed for the problem of representing incompletely specified Boolean and discrete functions by the Reed-Muller (RM) transform [1]. RM transform is a discrete function representation by a polynomial in n variables over a finite field. Although we are primarily interested in algorithms suitable for small finite fields, the algorithm can be used over any field.

In addition to the traditional applications over real numbers [2], interpolations over finite fields have been used for decoding error-correcting codes [3], testing [4], and in learning algorithms [5]. The application considered in our previous work [1] was that of obtaining an RM transform for functions with possibly many "don't care" points. It is known that the problem of finding such transforms, i.e., multivariate polynomials, with a bounded number of terms is NP-complete and the optimization approaches are exhaustive and time-consuming [6], [7]. The interpolation approach can instead produce the fitting multivariate polynomial of the size bounded by the number of specified points. Hence, the RM transform of an incompletely specified function can be obtained efficiently if a polynomial time algorithm can be devised for the interpolation problem. The same results then extend naturally to a broader class of polynomial representations [8], [9], as well as to RM-related graph-based circuit descriptions, such as Functional Decision Diagrams (FDDs) [10].

1.1 Multivariate Interpolation

The Lagrange or Newton interpolation algorithms can be used over any field to obtain the coefficients c_i of a univariate polynomial $f(x) = \sum_{i=0}^{t-1} c_i x^i$ of degree $t-1$ from the values at arbitrary t points. The problem, though, is much more difficult for multivariate functions. While the choice of terms $(1, x, x^2, \ldots, x^{t-1})$ and the result are unique in the univariate case, for multivariate functions, the term selection depends on the position of interpolation points. There is no known algorithm to select, in advance, the terms of a multivariate polynomial that guarantee the existence of a solution for an arbitrary set of points. Currently, it is possible to completely characterize the solutions for up to two-variable polynomials of degree at most four [11]. Due to the difficulty of the problem, most results on multivariate interpolation deal with somewhat relaxed cases.

1.1.1 Black Box Interpolation over Finite Fields

The "black box" interpolation model assumes that the algorithm can select the interpolation points *freely*. The degree of the polynomial is often given as an input. For this problem, the solution critically depends on the underlying field. Three cases need to be considered, depending on whether the field is $GF(2)$ [12], a small finite fields [13], or a large finite (and infinite) field [14]. For $GF(2)$, an effective procedure exists for selecting the points and solving the interpolation problem by decoding the Reed-Muller codes [12], [15]. For large fields, the algorithms [14] rely heavily on the size of the field. For small finite fields other than $GF(2)$, the solution exists only if the interpolation points are chosen from a sufficiently large extension field [18]. We refer to [16], [23], and [17] for further review.

1.1.2 Interpolation on Arbitrary Points

Much less is known about finding a solution when the interpolation points cannot be selected freely. One known n-variate, t point interpolation algorithm [19] requires a priori bound d on the degree in each variable to run in probabilistic $O(ndt^2)$ time. The algorithm is suited for larger fields as the probability of failure is quadratic in the number of points and decreases linearly with the field size. Substantial work on multivariate interpolation has been done in the area of numerical linear algebra. The algorithm by de Boor and Ron [2] attempts to solve the interpolation problem by Gaussian elimination. The algorithm by Sauer [20] tries to construct a set of the basis multivariate polynomials, similar to the Gram-Schmidt orthogonalization, to express the interpolating polynomial by a linear combination of the basis polynomials. Although both algorithms could possibly be used over finite fields, the polynomial run time is not guaranteed over any field.

This paper presents, in Section 2, an algorithm for multivariate polynomial interpolation that uses the structure of a multivariate generalized Vandermonde matrix to find suitable polynomial terms. The algorithm performance depends on the initial term selection, as shown in Section 2.3. In Section 3, we devise the decomposition of the interpolation into smaller problems over subspaces of the function definition domain. The decomposition alone is sufficient for the quadratic time $GF(2)$ interpolation, Section 3.3.

2 Deterministic Interpolation Algorithm

We are given t distinct points

$$p_1, p_2, \ldots, p_t \in GF(q)^n$$

and values

$$f_1, f_2, \ldots, f_t \in GF(q)$$

- *Z. Zilic is with the Department of Electrical and Computer Engineering, McGill University, 3480 University St., Montreal, Quebec H3A 2A7, Canada. E-mail: zeljko@macs.ece.mcgill.ca.*
- *Z.G. Vranesic is with the Computer Engineering Research Group, University of Toronto, Toronto, Ontario M5S 1A4, Canada. E-mail: zvonko@eecg.toronto.edu.*

Manuscript received 10 Mar. 2000; revised 10 Jan. 2002; accepted 25 Jan. 2002.
For information on obtaining reprints of this article, please send e-mail to: tc@computer.org, and reference IEEECS Log Number 111691.

that a function takes at these points. We want to fit a polynomial, that is a linear combination, $\sum_{j=1}^{t} c_j * M_j$, of at most t *terms (monomials)* $M_j = \prod_{l=1}^{n} x_l^{m_{jl}}$. Each term is specified by n exponents, m_{jl}, which are the integers in the range $0 \ldots q-1$. Hence, we will produce the coefficients

$$c_1, c_2, \ldots, c_t \in GF(q)$$

and the terms represented by the exponent vectors

$$m_1, m_2, \ldots, m_t \in \{0, 1, \ldots, q-1\}^n.$$

We use the lowercase letters, like p, to denote vectors of points, and indexed letters, as in p_i, for the individual points. For coordinates of points, we use double-indexed letters: p_{ij} indicates coordinate j of point p_i. The same convention is used for the terms.

Conceptually, the problem could be solved by a linear algebra approach. The coefficient vector c of a polynomial fitting the vector of values f can be obtained by solving the linear system $Tc = f$ if the matrix T is invertible, i.e., nonsingular.

Matrix T is the generalized multivariate Vandermonde matrix, obtained by applying the polynomial terms to the given t points, $T_{ij} = \prod_{l=1}^{n} p_{il}^{m_{jl}}$. We denote the entry T_{ij} with either $p_i^{m_j}$ or $m_j(p_i)$ as an application of a term to a point involves the coordinate-wise exponentiation. The rows of T correspond to the points and the columns correspond to the terms. We know that there always exist t terms for which the matrix T is nonsingular. (*Proof:* Consider a $t \times q^n$ matrix, obtained by applying all possible terms to the given points. This matrix has rank t, and, consequently, there exists a $t \times t$ submatrix of full rank.)

It is, however, not known how to make the matrix T nonsingular in advance by selecting the polynomial terms. Starting with arbitrary terms, our contribution is in exploiting the Vandermonde matrix structure during the algorithm execution such that the matrix is made nonsingular by a series of deterministic term replacements. The interpolation problem is then easily solved by matrix inversion.

2.1 Nullspaces and Increasing the Rank

The interpolation algorithm relies on the step of replacing a term in the interpolation polynomial, which deterministically increases the rank of the matrix T by one. For this, we use the nullspaces of the matrix. For a singular matrix, when the rank is not full, there exist constants $k_1, k_2, \ldots, k_t$ such that the following holds for each row i:

$$k_1 p_i^{m_1} + k_2 p_i^{m_2} + \ldots + k_t p_i^{m_t} = 0.$$

The vector $k = [k_1 k_2 \ldots k_t]$ belongs to the *column nullspace* C_{null} of matrix T. Such vectors form a nullspace whose dimension (*nullity*) is ν. Alternatively, since the row and column ranks are equal, we can consider the *row nullspace*, R_{null}. The vectors r and k in both nullspaces then satisfy

$$\sum_{i=1}^{t} r_i p_i^{m_j} = 0, \sum_{j=1}^{t} k_j p_i^{m_j} = 0; i, j = 1, 2, \ldots, t.$$

For our interpolation, we can freely choose only the terms of a polynomial and considering the row nullspace will help us select the terms. For a fixed row nullspace vector r, it is sufficient to find a term m_r for which $\sum_{i=1}^{t} r_i p_i^{m_r} \neq 0$ to remove that vector from the nullspace. The following theorem describes the replacement step.

Theorem 1. *The rank of T increases by 1 if a column v, corresponding to a nonzero component k_v of $k \in C_{null}$, is replaced by a column obtained by a term m_{t+1} for which $\sum_{i=1}^{t} r_i p_i^{m_{t+1}} \neq 0$, for $r \in R_{null}$.*

Proof. By adding a term m_{t+1} for which $\sum_{i=1}^{t} r_i p_i^{m_{t+1}} \neq 0$, we eliminate the row nullspace vector r. Then,

$$\sum_{i=1}^{t} r_i p_i^{m_j} = 0, j \neq t+1 \quad (1)$$

and

$$\sum_{i=1}^{t} r_i p_i^{m_{t+1}} \neq 0. \quad (2)$$

We can remove the column corresponding to a nonzero component k_v in the column nullspace vector k. Then, we obtain the nonzero linear combination

$$\sum_{j=1, j\neq v}^{t} k_j p_i^{m_j} = -k_v p_i^{m_v}. \quad (3)$$

We now claim that replacing the term m_v with m_{t+1} cannot possibly create a new column nullspace vector k'. Obviously, for nonzero sum in (3), the component k'_{t+1} is nonzero. Then, k' is a nullspace vector if

$$\sum_{j=1, j\neq v}^{t+1} k'_j p_i^{m_j} = 0$$

for each i. Multiplying each sum with r_i and adding them together also results in 0. This sum can be written as:

$$\sum_{i=1}^{t} r_i \sum_{j=1, j\neq v}^{t+1} k'_j p_i^{m_j} = \sum_{j=1, j\neq v}^{t+1} k'_j \sum_{i=1}^{t} r_i p_i^{m_j} = \sum_{j=1, j\neq v}^{t} k'_j \sum_{i=1}^{t} r_i p_i^{m_j} + k'_{t+1} \sum_{i=1}^{t} r_i p_i^{m_{t+1}}.$$

Using (1) and (2), this expression reduces to

$$k'_{t+1} \sum_{i=1}^{t} r_i p_i^{m_{t+1}} \neq 0$$

because $k'_{t+1} \neq 0$; this contradicts the assumption that k' is a null vector. Hence, replacing the column v by the column $t+1$ eliminates one nullspace vector and increases the rank only by 1 because we can always choose a basis of C_{null} which has only one nonzero coordinate k_v among all vectors k in the basis. Consequently, all the other base vectors will remain in C_{null} after this replacement step as they act on columns that did not change. □

This replacement step can be used in a deterministic polynomial interpolation algorithm. There can be at most t replacement steps. In each step, the nullspace vectors can be obtained in $O(t^3)$ time in a traditional way by either Gaussian elimination, Knuth's or Berlekamp's algorithm [21], or in $O(M(t))$ time, required for the fast matrix multiply [22] operation. However, it is not apparent that searching for the replacement term can be done in polynomial time because there are q^n possible terms to search from.

2.2 Efficient Search for Replacement Terms

We now show that the replacement term can be found in $O(nt^2)$ time by exploiting the structure of the multivariate Vandermonde matrix. We use the fact that we can obtain the following nullspace basis vector $r = [r_1 r_2 \ldots r_t]$. For such r, there is no nullspace vector r' with nonzero components that are a subset of the nonzero components of r. (*Proof:* Otherwise, a linear combination r'' of the two can eliminate the excess nonzero components, and the property will hold for r''.) We say that such a vector r is *coordinate reduced*.

Theorem 2. *Let vector $r \in R_{null}$ of generalized Vandemonde matrix be coordinate reduced. Among the existing terms, there is a term m_j with*

the following property: If a term m_{t+1} is created from m_j by increasing a coordinate m_{jd} by 1, then $\sum_{i=1}^{t} r_i p_i^{m_{t+1}} \neq 0$. It suffices to choose the coordinate d such that the point coordinates $p_{id}, i = 1, 2, \ldots, t$ are not constant for nonzero components of r.

Proof. First, there must exist a coordinate d for which p_{id} is not constant for all nonzero components r_i of r because, otherwise, some points would coincide.

For each existing term $m_l = [m_{l1} m_{l2} \ldots m_{ld} \ldots m_{ln}]$, we have

$$\sum_{i=1}^{t} r_i p_i^{m_l} = 0. \tag{4}$$

By way of contradiction, assume that there is no term m_j with the above property. Then, we obtain m'_l by increasing the coordinate m_{ld} by 1, i.e., $m'_l = [m_{l1} m_{l2} \ldots m_{ld} + 1 \ldots m_{ln}]$, and the expression

$$\sum_{i=1}^{t} r_i p_i^{m'_l} = \sum_{i=1}^{t} r_i p_{id} p_i^{m_l} = 0 \tag{5}$$

is true for $l = 1, 2, \ldots t$. By comparing (4) and (5), a vector with components $r'_i = r_i p_{id}$ would be a nullspace vector. This is impossible if the nullity ν is 1 because coordinates p_{id} are not constant for all is and these two vectors are not collinear. The dimension of the nullspace would then be 2, which contradicts our assumption.

When $\nu > 1$, then, in addition to the vector r' with coordinates $r'_i = r_i p_{id}$, we consider all linear combinations of $r'' = \alpha r + \beta r'$ that must be in the nullspace. Since sets of nonzero components of r and r' are equal and the r' and r are not collinear, there must exist r'' with nonzero components that are a subset of those in r. However, r'' cannot be a nullvector since the vector r is coordinate reduced, i.e., no nullspace vector has only a subset of its nonzero components. Hence, there must exist a term m_j with the desired property. □

This proof guarantees that the replacement term will be found among $O(t)$ alternatives, which is small relative to searching through all possible q^n terms.

The complete algorithm is given by pseudocode in Algorithm 1. The algorithm produces an $O(t)$ term interpolation polynomial using $O(t * M(t))$ field operations if the fast matrix multiply is applied or with $O(t^4)$ operations using standard linear algebra. The Vandermonde matrix construction step is also linear in n, but, since, normally, $t > n$ and the algorithm time is dominated by inversion steps that are independent of n, there is no overall running time dependence on n.

Algorithm 1: Interpolation by Vandermonde matrix nullspaces

- Select initial terms
- Create initial Vandermonde matrix T

$$T_{ij} = p_i^{m_j}; i, j = 1, \ldots, t$$

- If $det(T) \neq 0$, return terms and coefficients $T^{-1} * f$, else $\nu = nullity(T)$
- Repeat ν times
  - Obtain nullspaces, select $k_1 \in C_{null}$, and coordinate reduced $r_1 \in R_{null}$
  - Find d such that p_{id} is not constant for nonzero components of r_{1i}
  - Find term m_j for which

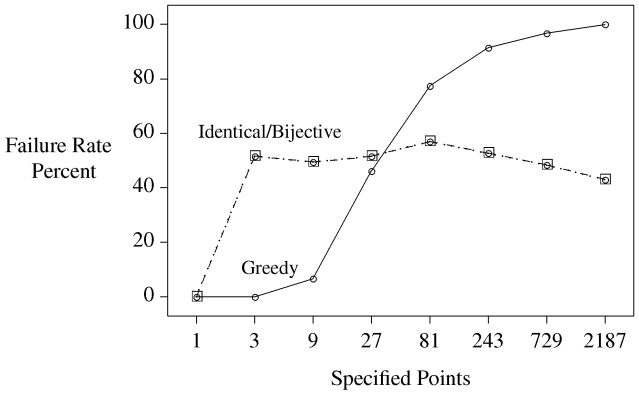

Fig 1. Initial failure rates for three selections of terms—GF(3).

$$\sum_{i=1}^{t} r_{1i} p_{id} * {p_i}^{m_j} \neq 0$$

  - Create m_{t+1} from m_j by adding 1 to coordinate m_{jd}
  - Replace column v for which $k_{1v} \neq 0$ with column generated by m_{t+1}
- Return polynomial terms $m_i, i = 1, \ldots, t$ and coefficients $c = T^{-1} * f$

2.3 Initial Selections of Term

Algorithm 1 starts with an initial selection of t terms and makes ν replacement steps. Although the algorithm will complete for any initial selection, having a large initial matrix rank would make the algorithm faster. The selection might also lead to polynomials that have large exponents, which are expensive to implement. In [23], we presented several heuristics for selecting the terms. They produce as small exponents as possible, while the initial rank is high. These selections are based on the following observation: If we consider any projections of points, then the number of terms in these projections should correspond to the maximum number of points. Otherwise, if there are more points than terms in some projection, there may exist no solution, even in the univariate case.

The simplest way to ensure that the number of points does not exceed the number of terms in any projection is by *identical selection*, i.e., having the terms equal to the points. A degree-reduced one-to-one mapping between point and term coordinates by which the most commonly used degrees in each variable are the lowest is called *bijective selection*. Finally, the *greedy selection* chooses the lowest degree terms such that the number of terms in the projection does not exceed the number of points. This goal is achieved by considering points one by one: For each new point, only the highest coordinate that has changed is increased by one. We proved in [23] that the greedy selection produces a nonsingular system for any three points in any field and for some configurations of n points in any field.

Since it is impossible to analyze fully even the case with two points, we compared the term selections on randomly generated incompletely specified functions over several small finite fields. Fig. 1 and Fig. 2 give the percentage of systems for which these initial term selections fail to produce a nonsingular system over $GF(3)$ and $GF(11)$, respectively. Since there is no apparent difference between the average ranks using the identical and bijective term selection schemes, the bijective term selection is preferred since it minimizes the degrees among all such term selections. Greedy selection works better for fewer points, while the bijective selection becomes better as the size of the field increases. The results for finite fields of order between 3 and 11, not shown here, show a gradual transition between the two cases

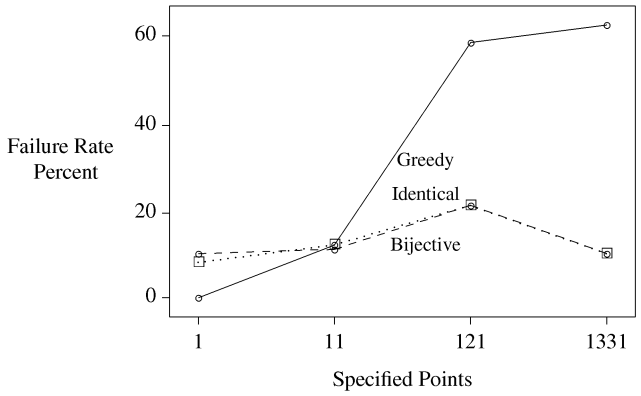

Fig. 2. Initial failure rates for three selections of terms—GF(11).

plotted. Comparing the final term degrees for each selection, we observe that they are bounded by the initial degrees plus the initial rank nullity, according to Theorem 2.

3 Problem Decomposition

By investigating the structure of the system $Tc = f$, we now decompose the original problem into smaller problems which can be solved independently. The decomposition is useful in speeding up the interpolation, and in parallel [24] and incremental [23] algorithms. The decomposition is most effective for small finite fields, which interest us the most.

3.1 Partial Ordering of Interpolation Spaces

We say that two points u and v are equivalent, $u \approx v$, if they have 0s in the same coordinates. This relation is an equivalence relation over the interpolation space and the equivalence classes are called *z-subspaces*. Since z-subspaces are distinguished only by coordinates which are 0, we denote them with expressions like S_{x0x0} to indicate which coordinates are zero and which have only nonzero values.

A relation of partial order $\preceq$ is defined between z-subspaces. A z-subspace S_1 precedes or equals S_2 if the set of coordinates that are 0 in S_2 is a subset of those in S_1. Incomparable z-subspaces are those whose zero coordinates form mutually noninclusive sets. We use the symbol $\parallel$ for the incomparability relation. Relations $\prec$ and $\succeq$ are defined using $\preceq$ and equality in a standard way.

Example 1. Points 1020 and 2010 belong to the z-subspace $S_1 = S_{x0x0}$. Points 1210 and 1012 belong to the z-subspaces $S_2 = S_{xxx0}$ and $S_3 = S_{x0xx}$, respectively. Since the sets of zero coordinates in both S_2 and S_3 are the subsets of zero coordinates in S_1, it follows that $S_1 \prec S_2$ and $S_1 \prec S_3$. Also, the latter two z-subspaces are not comparable, i.e., $S_2 \parallel S_3$, because the coordinate $S_{22} \neq 0$ while $S_{32} = 0$ and $S_{24} = 0$, but $S_{34} \neq 0$.

This relation maps the hypercube $(GFq)^n$ to the Boolean algebra B_n. We use this mapping to speed up the polynomial evaluation and decompose the interpolation. In our case, the poset representing the existing z-subspaces can be any subposet of B_n.

3.2 Structure of the System Matrix

The following statement holds:

Theorem 3. *An entry $m(p)$ of a matrix T is:*

1. *zero if $p \prec m$,*
2. *zero if m and p are not comparable,*
3. *nonzero if $p \succeq m$.*

Proof. The proof follows from the definition of relation $\prec$. Case 1) Some coordinate of p is zero, while it is nonzero in m, for which p^m results in a zero application p^m. Case 2) m and p have zero coefficients such that, in both $m(p)$ and $p(m)$, there is a term $e(0) = 0^e, e \neq 0$. Case 3), whenever a coordinate in p is 0, it is 0 in m as well and $0^0 = 1$. □

This characterization suffices to decompose the problem, as follows: When the z-subspaces of points and terms coincide, the system matrix is *block-triangular*.

Example 2. Let a sparse function be specified at z-subspaces S_{00x}, S_{0xx}, S_{x0x}, and S_{xxx}. Then, the matrix T consists of block matrices, each of which contains applications of terms from one z-subspace to points in another z-subspace. These block matrices consist either of all zeros or all nonzero elements, depending on the relative order between the point and term subspaces. The matrix takes the form

$$T = \begin{bmatrix} [0^0 0^0 x^x] & 0 & 0 & 0 \\ [0^0 x^0 x^x] & [0^0 x^x x^x] & 0 & 0 \\ [x^0 0^0 x^x] & 0 & [x^x 0^0 x^x] & 0 \\ [x^0 x^0 x^x] & [x^0 x^x x^x] & [x^x x^0 x^x] & [x^x x^x x^x] \end{bmatrix},$$

where each nonzero block matrix is represented by the values that the point and term coordinates can take. Note that, for each block, if there exists a coordinate in which 0 is raised to a nonzero coefficient, a block matrix filled with zeros is obtained.

Equivalent to this process is the inversion of block-triangular matrices. Another view is based on the geometric meaning of z-subspaces, which are the points, lines, planes, etc., in *n*-dimensional space. Fig. 3 shows the execution order of the algorithm for the three-dimensional case. The solution is first obtained for the z-subspace S_{000} (the point at origin), if present. Next, interpolations along lines S_{x00}, S_{0x0}, and S_{00x} can be obtained, followed by planes S_{0xx}, S_{x0x}, and S_{xx0}. The last z-subspace to be solved is in S_{xxx}. Since this order of execution is given by the poset of z-subspaces, we use that poset as a primary way to describe the interpolation by decomposition into z-subspaces.

An interpolation algorithm can be defined, based on the traversal of the poset of z-subspaces. After performing interpola-

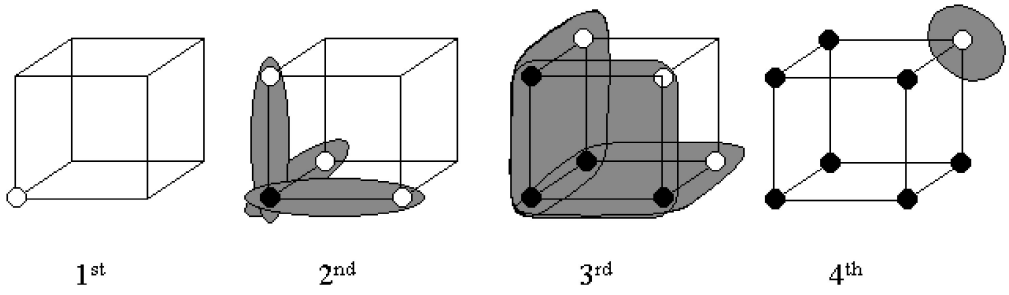

Fig. 3. Solving z-subspaces in order—newly found terms are in white dots, already known terms are in black.

TABLE 1
Three-Variable Partial Function

x_1	x_2	x_3	f
1	0	1	1
1	2	0	1
1	2	1	2
2	1	1	0
2	2	1	1

tion over each subspace, the algorithm evaluates the polynomial just obtained at all higher subspaces. Algorithm 2 allows us to select the terms independently for each subspace and even to change them during the execution because the overall matrix T is never constructed explicitly. All nondiagonal block matrices used in the derivation above need not be known in advance. The following example illustrates the proposed algorithm, including the decomposition into smaller problems in z-subspaces.

Algorithm 2: Interpolation using z-subspaces

- Sort z-subspaces according to $\prec$
- For each z-subspace S_i with points P_i and values f_i in increasing order
  - Select all polynomial terms M_i and create $T_i = [P_i^{M_i}]$
  - Interpolate in S_i to obtain a vector of coefficients

$$c_i = {T_i}^{-1} f_i$$

  - For all $S_j \succ S_i$, update vectors of values

$$f_j = f_j - [P_j^{M_i}] c_i$$

Example 3. Consider the three-variable partial function given by the points in $(GF(3))^3$ as shown in Table 1.

The first two points belong to the z-subspaces S_{x0x} and S_{xx0}, respectively, while the remaining three points belong to S_{xxx}. The ordering relation $\prec$ between these subspaces allows us to solve the first two subsystems independently. This gives the following two terms: $c_{101} = 1$ and $c_{120} = 1$. With these two coefficients, the system can be updated for the remaining points. We adjust the function values for points in S_{xxx} by subtracting the value of the calculated term at these points. This results in the value vector $[1, 1, 2]$ after the first coefficient is calculated and $[0, 2, 0]$ after both coefficients have been taken into account. The final step consists of setting up and solving a system of equations for points in S_{xxx}. The matrix T_{xxx} is constructed by calculating all possible values $p_j(p_i)$ (terms equal points):

$$T_{xxx} = \begin{bmatrix} 1 & 2 & 1 \\ 2 & 1 & 1 \\ 2 & 2 & 1 \end{bmatrix}.$$

After multiplying its inverse with the vector of values $[0, 2, 0]$, the last three coefficients are $c_{121} = 0$, $c_{211} = 1$ and $c_{221} = 1$. The result is: $f = x_1x_3 + x_1x_2^2 + x_1^2x_2x_3 + x_1^2x_2^2x_3$.

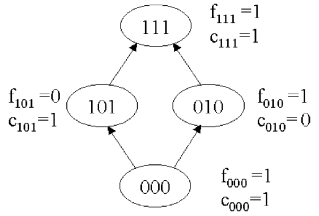

Fig. 4. GF(2) interpolation example using Algorithm 3.

The decomposition lowers the number of points that have to be considered at a time, but does not reduce the worst-case complexity as all the points can be in one z-subspace. It is easy to obtain the average-case performance: For uniform distribution of points, the largest z-subspace is of size $(1 - \frac{1}{q})^n * t$. The worst-case performance can be improved by using the shifted polynomials [1]. This decomposition is obviously most useful for small fields.

3.3 Binary Case

The decomposition of the interpolation just given can be applied to the binary case, where each z-subspace consists of a single point—consequently, the solution trivially exists. The $O(t^2)$ time algorithm is performed as in Algorithm 3.[1]

Algorithm 3: GF(2) Interpolation by Posets

- Sort points according to $\prec$
- For all bottom points $p_\perp$

$$c_\perp = f_\perp$$

- For all other points p_i in increasing $\prec$ order

$$c_i = f_i - \sum_{j=1}^{i-1} T_{ij} * c_j = f_i - \sum_{j<i} p_j(p_i) * c_j = f_i - \sum_{p_j \prec p_i} c_j \quad (6)$$

Example 4. Consider the poset diagram in Fig. 4. The function values f_p associated with a point p are used in the traversal to produce the coefficients c_p. For each term, equal to its point p, the polynomial coefficient is obained by subtracting the function value from the sum of all coefficents below the point, (6).

3.4 Algorithm Extensions

In [23], we presented three extensions of the proposed algorithm. An incremental version of the algorithm can be useful in logic synthesis and in learning algorithms. The problem decomposition, although not leading directly to provably parallel algorithms, can be very practical for many parallel machine models. The shifted polynomials (those whose variables are $x_1 + a_1, x_2 + a_2 \ldots x_n + a_n$) can be used to both speed up the algorithm and reduce the degree of the resulting polynomial.

4 Concluding Remarks

We presented a new algorithm for the multivariate interpolation problem in which the interpolation points are specified as inputs, rather than being selected freely. The algorithm can be used over

1. We believe that this special case was known before, although we found no direct reference to such an algorithm. An algorithm in [5] uses the poset structure in a similar way for GF(2) function learning, rather than for interpolation.

any field, although we applied it only to the finite field interpolation.

To the best of our knowledge, this is the first algorithm that solves the problem using the deterministic polynomial number of field operations. Compared to other algorithms over finite fields, the algorithm uses only tools of linear algebra, including the newly discovered properties of the generalized multivariate Vandermonde matrix. The algorithm deals directly with the nonsingularity of the system matrix. The longstanding openness of the problem attests to the difficulty of ensuring nonsingularity in advance. Unlike other approaches, which are probabilistic, restrict the scope of the problem and underlying fields, or abandon the polynomial runtime, we ensure the system matrix nonsingularity by constructing the polynomial term set during the algorithm execution. The algorithm starts with an initial selection of polynomial terms and then increases the matrix rank by a series of polynomial term replacements, based on the matrix nullspaces, until the matrix becomes invertible.

As a separate and independent contribution, we presented the decomposition of the multivariate interpolation, which is most useful for the small field case. By establishing a special type of the partial order relation between the interpolation points and polynomial terms, the original problem is transformed into smaller independent interpolation problems. We find that such a decomposition is useful in speeding up the serial algorithm and providing efficient incremental and parallel implementations.

ACKNOWLEDGMENTS

The authors thank Charles Rackoff, Frank Kschischang, Daniel Panario, and Allan Borodin from the University of Toronto for their help. Anonymous reviewers provided invaluable help in improving the final presentation of the paper.

REFERENCES

[1] Z. Zilic and Z.G. Vranesic, "A Multiple Valued Reed-Muller Transform for Incompletely Specified Functions," *IEEE Trans. Computers,* vol. 44, no. 8, pp. 1012-1020, Aug. 1995.

[2] C. de Boor and A. Ron, "Computational Aspects of Polynomial Interpolation in Several Variables," *Math. Computation,* vol. 58, pp. 705-727, 1992.

[3] V. Guruswami and M. Sudan, "Improved Decoding of Reed-Solomon and Algebraic-Geometric Codes," *Proc. Symp. Discrete Algorithms,* pp. 108-117, Nov. 1998.

[4] T. Damarla and M. Karpovsky, "Fault Detection in Combinational Networks by Reed-MullerTransforms," *IEEE Trans. Computers,* vol. 38, no. 6, pp. 788-797, June 1989.

[5] R.E. Schapire and L.M. Sellie, "Learning Sparse Multivariate Polynomials over a Field with Queries and Counterexamples," *Proc. Symp. Computational Learning Theory (COLT '93),* pp. 17-26, May 1993.

[6] D.H. Green, "Reed-Muller Expansions of Incompletely Specified Functions," *IEE Proc., Part E,* vol. 134, no. 5, pp. 228-236, Sept. 1987.

[7] A. Zakrevskij, "Minimum Polynomial Implementations of Systems of Incompletely Specified Boolean Functions," *Proc. Second Workshop Applications of the Reed-Muller Expansions in Circuit Design,* Aug. 1995.

[8] W.G. Schneeweiss, "On the Polynomial Form of Boolean Functions: Derivations and Applications," *IEEE Trans. Computers,* vol. 47, no. 2, pp. 217-221, Feb. 1998.

[9] K. Radecka and Z. Zilic, "Using Arithmetic Transform for Verification of Datapath Circuits via Error Modeling," *Proc. VLSI Test Symp. (VTS 2000),* pp. 271-277, 2000.

[10] Z. Zilic and K. Radecka, "Don't Care Minimization by Interpolation," *Proc. Int'l Workshop Logic Synthesis (IWLS '98),* pp. 353-356, May 1998.

[11] D. Bojanov, H.A. Hakopian, and A.A. Sahakian, *Spline Functions and Multivariate Interpolations.* Kluwer Academic, 1993.

[12] A. Dur and J. Grabmeier, "Applying Coding Theory to Sparse Interpolation," *SIAM J. Computing,* vol. 22, no. 4, pp. 695-703, Aug. 1993.

[13] D.Y. Grigoriev, M. Karpinski, and M.F. Singer, "Fast Parallel Algorithms for Sparse Multivariate Polynomial Interpolation over Finite Fields," *SIAM J. Computing,* vol. 19, no. 6, pp. 1059-1063, Dec. 1990.

[14] M. Ben-Or and P. Tiwari, "A Deterministic Algorithm for Sparse Multivariate Polynomial Interpolation," *Proc. 20th Symp. Theory of Computing,* pp. 301-309, Apr. 1988.

[15] R.M. Roth and G.M. Benedek, "Interpolation and Approximation of Sparse Multivariate Polynomials over GF(2)," *SIAM J. Computing,* vol. 20, no. 2, pp. 291-314, Apr. 1991.

[16] E. Kaltofen, W.-S. Lee, and A.A. Lobo, "Early Termination in Ben-Or/Tiwari Sparse Interpolation and a Hybrid of Zippel's Algorithm," *Proc. Int'l Symp. Symbolic and Algebraic Computing*, pp. 192-201, 2000.

[17] K. Werther, "The Complexities of Sparse Polynomial Interpolation over Finite Fields," *Applicable Algebra in Eng., Comm., and Computing,* vol. 5, pp. 192-201, 1994.

[18] M. Clausen, A. Dress, J. Grebmeier, and M. Karpinski, "On Zero-Testing and Interpolation of k-Sparse Polynomials over Finite Fields," *Theoretical Computer Science,* vol. 84, no. 2, pp. 151-164, Jan. 1991.

[19] R. Zippel, "Interpolating Polynomials from Their Values," *J. Symbolic Computation,* vol. 9, pp. 375-403, Mar. 1990.

[20] T. Sauer, "Polynomial Interpolation of Minimal Degree," *Numerische Mathematik,* vol. 78, pp. 59-85, 1997.

[21] D. Knuth, *The Art of Computer Programming, second ed., vol. 2: Seminumerical Algorithms.* Reading, Mass.: Addison-Wesley, 1980.

[22] J. von zur Gathen and G. Juergen, *Modern Computer Algebra.* Cambridge Univ. Press, 1999.

[23] Z. Zilic, "Towards Spectral Synthesis: Field Expansions for Partial Functions and Logic Modules for FPGAs," PhD dissertation, Dept. of Electrical and Computer Eng., Univ. of Toronto, Jan. 1997.

[24] Z. Zilic and Z. Vranesic, "Parallel Sparse Finite Field Interpolation," *Proc. First IEEE Workshop Randomized Parallel Computing,* pp. 9-16, Apr. 1996.