On Feasible Multivariate Polynomial Interpolations over Arbitrary Fields

Zeljko Zilic Katarzyna Radecka

McGill University, Dept. of Electrical and Computer Engineering
3480 University St., Montréal, Québec H3A 2A7, Canada

Abstract

In this paper, we consider a problem of interpolating a multivariate polynomial from its values at arbitrary t points over a field F. We derive a deterministic algorithm that finds an interpolating polynomial with at most t terms. Relative to the univariate interpolation, minimal degree selection of terms and uniqueness cannot be guaranteed. Our construction uses the nullspaces of the multivariate generalized Vandermonde matrix associated with the problem to make this matrix nonsingular in a series of steps. The structure of this matrix allows us to deterministically find the terms that increase the rank of this matrix. We present a practical algorithm for finite field interpolations, together with a set of heuristics for obtaining fast a small-degree interpolation polynomial. As a special case of interpolation algorithm, we propose the quadratic time algorithm for interpolation over $GF(2)$ field.

1 Introduction

Interpolations have many applications in areas such as symbolic [14] and numerical computing [11], [4] decoding of the error-correcting codes [7], learning theory [6], etc.. In this paper, we consider a problem of interpolating a multivariate polynomial from its values at arbitrary t points over a field F, applied originally to synthesis of logic circuits [13]. We show that an interpolating polynomials with at most t terms can be found within a polynomial number of field operations. Further, we provide practical algorithms for interpolations over finite fields $GF(q)$ and describe their efficient implementations.

1.1 Multivariate Interpolation

The Lagrange or Newton interpolation algorithms can be used over any field to obtain 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 is much more difficult for multivariate functions. While the selection of terms $(1, x, x^2, \ldots, x^{t-1})$ and the result are unique in the univariate case, in the multivariate case 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 characterize only up to 6 points for which an interpolation exists among the 2-variable polynomials of degree four [2]. Because of the difficulty of the problem, most results on multivariate interpolation deal with somewhat relaxed problems.

1.1.1 Black box interpolation

The "black box" interpolation has been the most studied multivariate interpolation problem. In this model it is assumed that the algorithm can select the interpolation points *freely*, and the degree of the polynomial is often given as an input. Three cases need to be considered, depending if the fields are $GF(2)$ [5], small finite fields [3], or large finite (and infinite) fields [1]. For $GF(2)$, an effective procedure exists for selecting the point by solving the interpolation problem by decoding error-correcting codes [5], [10]. For large fields, the algorithms in [1] and [8] rely heavily on the large size of the field. The algorithm [1] requires computations with very large numbers that can occur in the process. A randomized algorithm for interpolations over large, but finite, fields was derived in [8] from this algorithm. For small finite fields (other than $GF(2)$) the solution exists only if the interpolation points are chosen from a sufficiently large extension field [3].

1.1.2 Interpolation on arbitrary points

Much less is known about finding a solution when the interpolation points cannot be selected freely by the algorithm. The known multivariate interpolation algorithm is probabilistic, and requires larger fields [14], as the probability of failure is quadratic in the number of points and decreases linearly with the field size. The algorithm uses univariate interpolations to fit the one-dimensional projections of the interpolation points. It first finds a polynomial in one variable, e.g. x_1, at some random assignment of other variables $x_2, x_3 \ldots, x_n$. When this univariate polynomial $\sum_0^{t_1} c_i * x_1^i$ is found, (t_1 depends on the selected points, and is bounded by a number of points t), then, by repeating the procedure, the coefficients c_i are expressed as a function of variables $x_2, x_3, \ldots, x_n$ in subsequent steps.

Much work on multivariate interpolation has been done in the area of numerical linear algebra. The algorithm by

de Boor and Ron [4] tries to solve the interpolation problem by Gaussian elimination. The algorithm by Sauer [11] tries to construct a set of the basis multivariate polynomials, similar to the Gram-Schmidt orthogonalization. The resulting polynomial is then expressed by a linear combination of the basis polynomials. Although both algorithms could possibly be used for interpolations over any field, including finite fields, the polynomial running time cannot be guaranteed. Interestingly, the most cumbersome aspect of these two algorithms involves the symbolic manipulation of the basis polynomials used towards constructing the solution.

Section 2 proves the existence of an algorithm for multivariate polynomial interpolation that requires a polynomial number of field operations. The proof uses the structure of a multivariate generalized Vandermonde matrix to find suitable terms. Next, an efficient implementation the finite field interpolation is described in Section 3. Based on the partial order between the interpolation points, we show in Section 3.1 how the interpolation problem can be decomposed into interpolations over suitable subspaces of the function definition domain. The decomposition alone is sufficient to achieve the quadratic time $GF(2)$ interpolation.

2 Existence of a Deterministic Interpolation Algorithm

We are given t distinct points

$$p_1, p_2, \ldots, p_t \in F^n$$

and values

$$f_1, f_2, \ldots, f_t \in F$$

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)* $M_j = \prod_{k=1}^{n} x_k^{m_{jk}}$. Each term is specified by n exponents, m_{jk}, which are the integers within a range. For finite fields, each of the exponents is in the range $0 \ldots q-1$. Otherwise, by considering the interpolation algorithm in [15] each of the exponents can be bounded by $t-1$. Hence, we will produce the coefficients

$$c_1, c_2, \ldots, c_t \in F$$

and the terms represented by the exponent vectors

$$m_1, m_2, \ldots, m_t$$

We use the lower-case letters, like p, for 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}, denotes coordinate j of point p_i. The same convention is used for the terms.

A solution can in principle be obtained by solving the linear system

$$Tc = f$$

which corresponds to the condition that the given vector of values f and the sought polynomial coincide at the points given. Vector c then contains the coefficients of the polynomial.

Matrix T is the generalized multivariate Vandermonde matrix, obtained by applying the terms to the given t points, $T_{ij} = \prod_{k=1}^{n} p_{ik}^{m_{jk}}$. We will 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 correspond to the points and the columns correspond to the terms. There always exist t terms for which the matrix T will be nonsingular. (*Proof:* Consider a matrix obtained by applying all the terms that can be in a solution, to the given points. For finite fields, this matrix is of size $t \times q^n$, and for infinite fields, it is sufficient to consider a finite matrix with $t \times t^n$ entries. This matrix has rank t, and consequently there exists a $t \times t$ submatrix of full rank.)

It is not known in advance how to select the terms to make the matrix of the system nonsingular. We start with arbitrary terms and in subsequent steps replace some terms until the matrix becomes nonsingular. Then, an inverse exists and the interpolation problem can be solved by inverting this matrix. The higher the rank of the starting matrix is, the fewer such replacements are needed; in the worst case, $O(t)$ such replacements are sufficient.

2.1 Nullspaces and Increasing the Rank

The interpolation will be possible if the matrix T is nonsingular after successively replacing terms in the interpolation polynomial. We show that the rank of T can be deterministically increased by considering its nullspaces.

When the rank is not full, then there exists a linear combination of columns (or rows) which is zero. Concentrating on rows, the following condition holds for each row i:

$$c_1 p_i^{m_1} + c_2 p_i^{m_2} + \ldots + c_t p_i^{m_t} = 0$$

, where vector $c = c_1 c_2 \ldots c_t$ belongs to the *column nullspace* C_{null} of matrix T. All such vectors are linearly independent, as they form a basis for the 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 c in both nullspaces then satisfy

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

For our purposes, 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. We now prove that such a replacement creates a matrix T whose rank increases by one.

Theorem 1 *The rank of T increases by 1 if a column v, corresponding to a nonzero component c_v of $c \in C_{null}$ 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

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

Now we show that we can remove the column corresponding to a nonzero component c_v in the column nullspace vector c. Then, we obtain the nonzero linear combination

$$\sum_{j=1, j\neq v}^{t} c_j p_i^{m_j} = -c_v p_i^{m_v} \tag{3}$$

We now claim that replacing the term m_v with m_{t+1} cannot possibly create a new column nullspace vector c'. For this to be true, the component c'_{t+1} must be nonzero; otherwise, the sum from Equation 3 is nonzero. Then, c' is a nullspace vector if

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

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

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

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

Using Equations 1 and 2 this expression reduces to

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

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

This replacement 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 Knuth's or Berlekamp's algorithm [9], or in $O(M(t))$ time, required for the fast matrix multiply operation. However, it is not apparent that searching for the replacement term can be done in polynomial time, because there are $O(t^n)$ (q^n in the case of finite fields) possible terms to search from.

2.2 Searching for Replacement Terms Efficiently

We will show that the replacement term can be found in $O(nt^2)$ time by using the structure of the multivariate Vandermonde matrix. We will use a fact that a basis vector $r = r_1 r_2 \dots r_t$ in the nullspace can be obtained with the following property. 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 $r \in R_{null}$ of generalized Vandemonde matrix be coordinate reduced. Among the existing terms, there exists a term m_j with the following property. If a term m_{t+1} is created from m_j by changing a coordinate m_{jd} by 1, then $\sum_{i=1}^{t} r_i p_i^{m_{t+1}} \neq 0$. It suffices to choose d such that the point coordinates $p_{id}, i = 1, 2, \dots, t$ 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 $m_k = m_{k1} m_{k2} \dots m_{kd} \dots m_{kn}$, we have

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

By way of contradiction assume that there is no term m_j with the above property. Then, we obtain m'_k by increasing the coordinate m_{kd} by 1, i.e. $m'_k = m_{k1} m_{k2} \dots m_{kd} + 1 \dots m_{kn}$, and the following expression

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

will be true for $k = 1, 2, \dots t$. 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 i's, and these two vectors are not collinear. The dimension of the nullspace would then be 2, which contradicts our assumption.

When $\nu > 1$, 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, then r'' will have 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. ■

Algorithm 1 *Interpolation by Vandermonde matrix nullspaces*

- *Find initial terms*
- *Create initial matrix* T

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

- *If* $det(T) \neq 0$ *return terms and coefficients* $T^{-1} * f$, *else* $\nu = nullity(T)$
- *Repeat* ν *times*
  - *Obtain coordinate reduced* $r_1 \in R_{null}$, $c_1 \in C_{null}$
  - *Find* d *such that* p_{id} *is not constant for nonzero components* r_{1i}
  - *Find term* m_j *for which*

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

  - *Create* m_{t+1} *from* m_j *by adding 1 to coordinate* m_{jd}
  - *Replace column* v *for which* $c_{1v} \neq 0$ *with column generated by* m_{t+1}
- *Return terms and coefficients* $T^{-1} * f$

The replacement term will be found among $O(t)$ alternatives. The Algorithm 1 can perform such an interpolation over any field.

The algorithm produces an interpolation polynomial with t terms using $O(t * M(t))$ field operations when the fast matrix multiply is used, and with $O(t^4)$ operations using standard linear algebra. We assume that $t > n$.

3 Interpolation Over Finite Fields

The implementation of Algorithm 1 can lead to an efficient algorithm for interpolation over finite fields. In that case, the operations over the Vandermonde matrix can be performed exactly by a finite-word machine. Further optimizations will be presented for an important case of small finite fields, which are important in error-correcting codes and in the applications of synthesizing Boolean and discrete functions [13].

3.1 Problem Decomposition

By considering the structure of the system $Tc = f$, we show that the original problem can be decomposed into several smaller problems which can be then solved independently. While the decomposition can be used for any field, it is most effective for small finite fields, which interest us the most.

3.1.1 Partial Order and Interpolation Subspaces

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 non-inclusive sets. We use the symbol $\|$ 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.* $S_2 \| S_3$, *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 $(GF(q))^n$ to the Boolean algebra B_n. We will use this mapping to speed up the polynomial evaluation and decompose the interpolation. In our case, for sparse functions, the poset representing the existing z-subspaces can be any sub-poset 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:*
i) zero if $p \prec m$
ii) zero if m *and* p *are not comparable*
iii) nonzero if $p \succeq m$

Proof: The proof follows from the definition of relation $\prec$. Case *i)* Some coordinate of p is zero, while it is nonzero in m, for which p^m results in a zero application p^m. Case *ii)* 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 *iii)* 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 nonzero coefficient, a block matrix filled with zeros is obtained.

The singularity condition for matrix T can be extended to the algorithm that uses the decomposition.

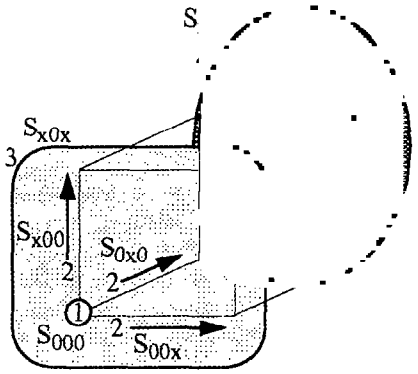

Figure 1: Solving z-subspaces in order - geometric presentation for three variables

Theorem 4 *A system $Tc = f$ is singular if and only if there are sets of points $P_1, P_2, \ldots P_z$ each completely contained in z-subspaces $S_1, S_2, \ldots S_z$, $z > 0$ where each z-subspace considered in isolation contains a singular system.*

Proof: (=>) Suppose that the matrix is singular. This requires that some of the rows sum to zero (*anull*). It is impossible for this to happen when we apply terms to points from incomparable subspaces. If such points are in z-subspaces S_1 and S_2, then the matrix T will be block diagonal and singular if at least one of the blocks is singular.

It is impossible that the z-subspaces that are comparable but not equal, $S_1 \prec S_2$, will anull, while those over S_2 and S_1 alone will not. We could then solve the system for S_1 and obtain the system defined only over S_2 by subtracting the computed values.

(<=) Conversely, if at least one subspace contains a singular system, then solvability of this subspace will not be improved if additional subspaces are taken into account. ■

Since only points in the same z-subspace can create a singular system, the term search steps will only be performed in these subspaces. 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. Figure 1 shows the execution order of the algorithm for the 3-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 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 interpolation over each subspace, the algorithm evaluates the polynomial just obtained at all higher 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*
    $$f_j = f_j - [P_j^{M_i}]c_i$$

This algorithm allows selecting the terms independently for each subspace and even changing them during the execution because the overall matrix T is never constructed explicitly. All non-diagonal block matrices used in the derivation above need not be known in advance. The following three-valued example illustrates the proposed algorithm, including the decomposition into smaller problems in z-subspaces.

Example 3 *Consider the three-variable partial function given by the points in $(GF(3))^3$:*

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

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 resulting polynomial is:

$$f = x_1x_3 + x_1x_2^2 + x_1^2x_2x_3 + x_1^2x_2^2x_3.$$

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

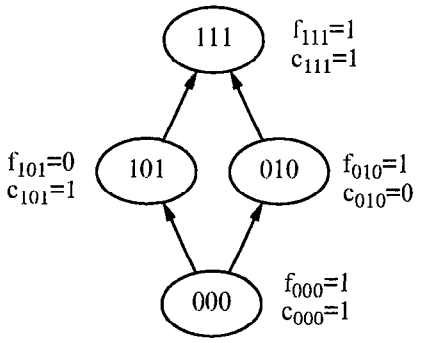

Figure 2: GF(2) interpolation example using Algorithm 3; function values and coefficients are shown next to their points.

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 [13]. This decomposition is obviously most useful for interpolations over small fields.

3.3 Binary Case

The decomposition of the interpolation problem can be applied to the binary case, where each z-subspace consists of a single point - consequently, the solution trivially exists. The polynomial terms selected will be equal to the points. The $O(t^2)$ time algorithm will be performed as follows.

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 function given by the poset diagram in Figure 2. The function values* f_p *associated with each point* p *are used in the traversal to produce the coefficients* c_p *for each term (equal to its point* p*).*

3.4 Optimizations and Extensions

3.4.1 Initial Selections of Terms

The algorithm starts with an initial selection of t terms and makes ν replacement steps. Clearly, having a large rank initial matrix would make the algorithm fast. In [12] we presented several heuristics for selecting the terms. They produce as small exponents as possible, while the rank is high. These selections are based on the following observation. If

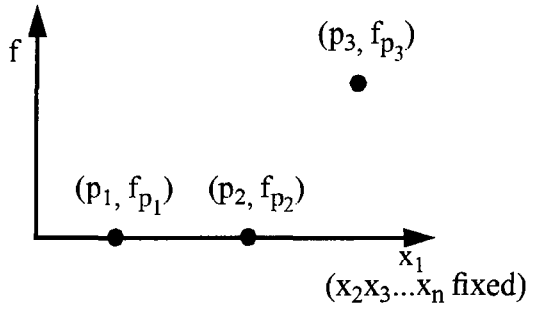

Figure 3: More points than terms in projection - example

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. Figure 3 shows an example, in which a one-dimensional projection contains three points. Selecting the polynomial with two terms in x_1 might not be sufficient. Hence, the number of points in any projection defines the minimal degree of the polynomial in the variables determined by that projection.

3.4.2 Bijective selection of terms

If we select each coordinate of the term set by some bijective mapping from the coordinates in the point set, then the number of points and terms in each projection will be the same. The cost can be minimized by selecting the *minimum degree terms*, by assigning the lowest degrees to the most common point coordinates. The algorithm can be stated as follows:

Algorithm 4 *Bijective Term Selection*

- *For each coordinate* $i = 1 \ldots n$
  - CS_i *= set of i-th coordinates of all points*
  - *Order the elements of* CS_i *according to how often they appear*
  - *For each point* p *replace the coordinate* p_i *with its position in the ordered* CS_i

While this produces an optimal bijective selection with respect to the total cost of the terms that might be used, the optimality cannot be guaranteed for the actual forms since it is not known which of these terms will be multiplied by nonzero coefficients. Note that the related t-Interpolation Decision Problem is NP-complete even in the binary case [10]. Such selection works well for large fields and point sets, as will be shown later, but for small fields, it can still often fail to produce a nonsingular system. resents one such case.

3.4.3 Greedy term selection

For small fields and few points, another approach for selection of terms can lead to a nonsingular system, while producing even less costly terms. It does not preserve the projections; on the contrary, it tends to increase many of them.

Starting from the initial point $11 \ldots 1$ in a z-subspace (0s are ignored since they will not change), a coefficient in only

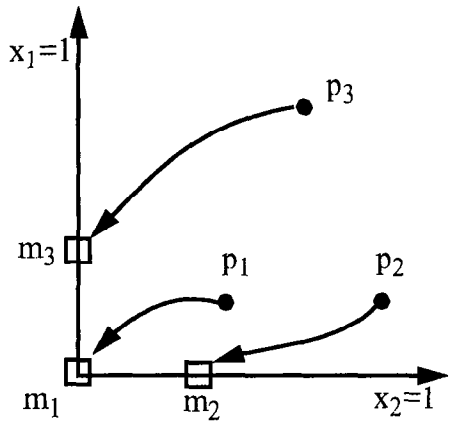

Figure 4: Example of greedy term selection

one direction that has changed is increased. The coordinate to increase can be chosen, for example, by lexicographically ordering the points. Intuitively, this corresponds to "greedy compaction" of the points, which lowers the coordinates of the chosen points each time there is an empty place where the point can be moved. The highest coordinate changed will increase by 1, while all lower ones will be set to one. This mapping can be done in linear time if the points are ordered.

Algorithm 5 *Greedy Term Selection*

- *Order the points lexicographically*
- *For each point*
  - *the highest coordinate that has changed is increased by 1,*
  - *all of the lower order coefficients are set to 1 (zeros ignored).*

Figure 4 shows an example of such term selection. The terms are "compacted", which is good for reducing the cost of the resulting form. For many practical examples, the greedy selection can increase the rank of the initial matrix while keeping the overall degree lowest among the heuristics.

The greedy selection of terms produces a nonsingular system for any three points in any field, and for some configurations of n points in any field. The proofs of these claims are given in [12].

Since it is impossible to analyze even the case with two points in full detail, we bring some empirical evidence on the usefulness of the term selection schemes.

We compared their performance on interpolations for logic synthesis benchmark functions over several small finite fields. Figures 5 and 6 give the percentage of systems for which these term selections fail to produce a nonsingular system, for functions over $GF(3)$ and $GF(11)$, respectively. The x axis on the graphs does not show the actual number of points. Since the scale is logarithmic, the highest power of the field size smaller than the actual number of points is shown. Since there is no apparent difference between the average performance of the identical and bijective term selection schemes, we conclude that the bijective term selection is preferred, because it minimizes the degrees among all such

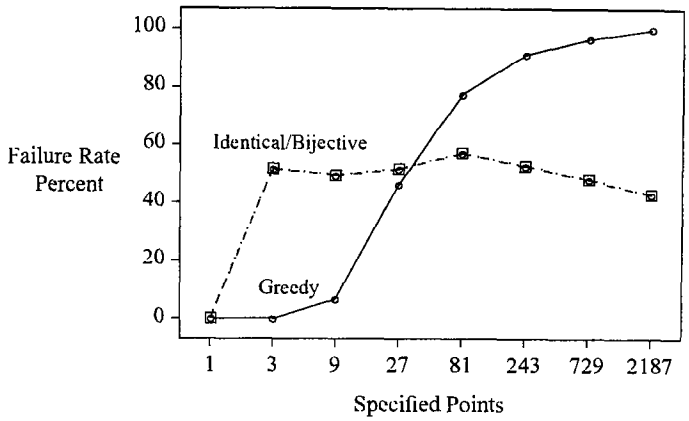

Figure 5: Failure rates for three selections of terms - GF(3)

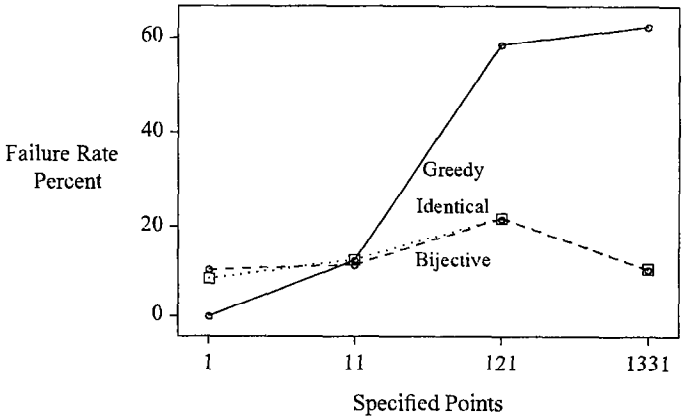

Figure 6: Initial failure rates for three selections of terms - GF(11)

term selection.

The 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 the gradual transition between the two cases plotted.

3.4.4 Algorithm Extensions

In [12] 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

In this paper we presented a new algorithm for multivariate interpolation. The deterministic algorithm increases the rank of the associated generalized Vandermonde matrix by a series of term replacements, based on its nullspaces. The main step of the algorithm can be used over any field, although a practical algorithm was derived only for finite field

interpolation. We also presented the decomposition of the problem which is the most useful for the small field case.

Acknowledgments: We thank Charles Rackoff, Frank Kschischang, Daniel Panario and Allan Borodin from the University of Toronto for their help.

References

[1] BEN-OR, M., AND TIWARI, P. A deterministic algorithm for sparse multivariate polynomial interpolation. *20th Symposium on Theory of Computing* (April 1988), 301–309.

[2] BOJANOV, D., HAKOPIAN, H. A., AND SAHAKIAN, A. A. *Spline Functions and Multivariate Interpolations.* Kluwer Academic Publishers, 1993.

[3] CLAUSEN, M., DRESS, A., GREBMEIER, J., AND KARPINSKI, M. On zero-testing and interpolation of k-sparse polynomials over finite fields. *Theoretical Computer Science 84*, 2 (January 1991), 151 164.

[4] DE BOOR, C., AND RON, A. Computational aspects of polynomial interpolation in several variables. *Mathematics of Computation 58* (1992), 705–727.

[5] DUR, A., AND GRABMEIER, J. Applying coding theory to sparse interpolation. *SIAM Journal of Computing 22*, 4 (August 1993), 695–703.

[6] GOLDREICH, O., RUBINFELD, R., AND SUDAN, M. Learning polynomials with queries: The highly noisy case. Report, Electronic Colloqium on Computational Complexity, August 1998.

[7] GURUSWAMI, V., AND SUDAN, M. Improved decoding of reed-solomon and algebraic-geometric codes. In *Symposium on Discrete Algorithms* (Palo Alto, California, November, 1998), ACM-SIAM, pp. 108 117.

[8] HUANG, M.-D. A., AND RAO, A. J. Interpolation of sparse multivariate polynomials over large finite fields. In *Symposium on Discrete Algorithms* (Altanta, Georgia, January, 1996), ACM-SIAM, pp. 508–517.

[9] KNUTH, D. *The Art of Computer Programming, 2nd Ed. Vol. 2: Seminumerical Algorithms.* Addison-Wesley, Reading, Massachusetts, 1980.

[10] ROTH, R. M., AND BENEDEK, G. M. Interpolation and approximation of sparse multivariate polynomials over GF(2). *SIAM Journal of Computing 20*, 2 (April 1991), 291–314.

[11] SAUER, T. Polynomial interpolation of minimal degree. *Numerische Mathematik Manuscript* (to appear in1996).

[12] ZILIC, Z. Towards spectral synthesis: Field expansions for partial functions and logic modules for FPGAs. Phd dissertation, University of Toronto, Department of Electrical and Computer Engineering, January 1997.

[13] ZILIC, Z., AND VRANESIC., Z. G. A multiple valued reed-muller transform for incompletely specified functions. *IEEE Transactions on Computers 44*, 8 (Aug. 1995.), 1012 1020.

[14] ZIPPEL, R. Interpolating polynomials from their values. *Journal of Symbolic Computation 9* (March 1990), 375-403.

[15] ZIPPEL, R. *Effective Polynomial Computation.* Kluwer Academic Publishers, 1993.