Journal of Mathematical Research & Exposition
Vol.19, No.4, 659-666, November 1999

An Algorithm for Polynomials That Commute with a Permutation Polynomial over a Finite Field *

Chong-Yun Chao[1], *Hong Zhang*[2]

(1. Dept. of Math., University of Pittsburgh, Pittsburgh, PA 15260;
2. Dept. of Math. Sci., Indiana-Purdue University, Fort Wayne, IN 46805)

Abstract: An efficient algorithm for determining polynomials that commute with a permutation polynomial over a finite field is given. The complexity of the algorithm is discussed and examples of applying the algorithm are also provided.

Key words: algorithm; polynomial; finite field.

Classification: AMS(1991) 12E/CLC O153.4

Document code: A **Article ID:** 1000-341X(1999)04-0659-08

1. Introduction

Let p be prime, n be a positive integer, and $F_q = GF(p^n)$ be the Galois field with $q = p^n$ elements. A polynomial $f(x)$ over F_q is said to be a permutation polynomial if $f(x)$, considered as a function from F_q to F_q, is a bijection.

Given a permutation polynomial $f(x)$ over F_q, we consider the problem of determining all polynomials (or all permutation polynomials) $g(x)$ over F_q such that $f(g(x)) = g(f(x))$, for all x in F_q, i.e., g commutes with f as functions on F_q.

In [5], Wells characterized the polynomials that commute with a translation $f(x) = x+a$. In [4], Mullen characterized the polynomials that commute with a linear permutation polynomial $f(x) = ax + b$. In [2], the general problem of determining polynomials that commute with an arbitrary permutation polynomial $f(x)$ was studied based upon the concept of centralizer rings by Schur.

Here we present an efficient algorithm and its implementation to determine the set of polynomials that commute with a given permutation polynomial over a finite field. A variation of the algorithm can be used to determine the set of permutation polynomials that commute with a given permutation polynomial. A detailed complexity analysis of the algorithm is given. It is compared to the complexity of the "brute force" appoach. By using the algorithm some examples are also provided.

Our algorithm is based on the results given in [2]. For convenious, we give a summary of the main results in [2].

*Received date: 1997-03-12

Let $M_{n\times n}$ be the set of all $n \times n$ matrices that on each row there is exactly one 1 and all other entries are 0. Let $D_n = \{0, 1, \cdots, n-1\}$. To each function $f : D_n \to D_n$, there is a corresponding matrix $A_f = (a_{i,j})$ in $M_{n\times n}$ with $a_{i,f(i)} = 1$ for $i = 0, 1, 2, \cdots, n-1$ and all other entries 0.

Let G be a permutation group acting on D_n and R be a ring with identity. For a permutation σ in G, let P_σ be the permuation matrix corresponding to σ. The centralizer ring $C_R(G)$ is defined as

$$C_R(G) = \{E; E \text{ is an } n \times n \text{ matrix over } R, \text{ and } P_\sigma E = EP_\sigma \text{ for every } \sigma \in G\}$$

Then it is proven in [2] that a function g in $\mathcal{F}$ commutes with σ, for every $\sigma \in G$ if and only if $A_g \in C_R(G) \cap M_{n\times n}$. As a corollary, the set of functions commuting with a single permutation σ corresponds to the set of matrices $C_R(G) \cap M_{n\times n}$, where G is the cyclic group generated by σ.

To construct a basis for the R-module $C_R(G)$, we define a relation $\sim$ on the set $D_n \times D_n$ as follows: $(i, j) \sim (r, s)$ if and only if there exists a σ in G such that $i\sigma = r$ and $j\sigma = s$. Since G is a group, this relation is an equivalence relation on $D_n \times D_n$. Denote the equivalence classes by $E_1, E_2, \cdots, E_k$ and corresponding to each $E_t, 1 \le t \le k$, define an $n \times n$ matrix $B_t = (b^t_{i,j})$ by

$$b^t_{ij} = \begin{cases} 1, & \text{if } (i,j) \in E_t, \\ 0, & \text{otherwise.} \end{cases}$$

It is shown that $\{B_1, B_2, \cdots, B_k\}$ is a basis of the R-module $C_R(G)$.

This provides a description of the structure of the functions that commute with the permutation in terms of matrices. It also gives a way to enumerate such functions. A permutation is said to be of type $(n_1, n_2, \cdots, n_n)$ if in the cycle decomposition the number of k-cycles is n_k. Let the permutation cooresponding to $f(x)$ be of type $(n_1, n_2, \cdots, n_q)$. The number of functions that commute with $f(x)$ is

$$\prod_k (\sum_{j|k} j n_j)^{n_k}.$$

The number of permutation functions that commute with $f(x)$ is

$$\prod_{k=1}^{n} (n_k!(k^{n_k})).$$

2. An algorithm

Based on the results in [2], we may devise an efficient algorithm to obtain all polynomials commuting with a give permutation polynomial.

Step 1. Evaluate the polynomial $f(x)$ to obtain a permutation σ on F_q.

Step 2. Find the cycle decomposition of σ, the permuation τ that has the same cycle structure as σ with its cycles arranged in increasing order. Determine the permutation μ that maps the symbols of σ to τ, i.e., $\sigma = \mu^{-1}\tau\mu$.

Step 3. Construct the matrix that represents the centralizer ring

$$\begin{bmatrix} A_{1,1} & 0 & \cdots \\ A_{2,1} & A_{2,2} & \cdots \\ & \cdots & \end{bmatrix},$$

where the matrix is partitioned according to the lengths of the cycles in τ. $A_{i,j}$ are circulant matrices and if the number of columns does not divide the number of rows $A_{i,j} = 0$. Determine the functions $h_1, h_2, \cdots, h_k$ that commute with τ by choosing in the above matrix one 1 entry in each row.

Step 4. Apply the mapping $g_i(x) = \mu^{-1}(h_i(\mu(x)))$ to obtain functions that commute with σ.

Step 5. Interpolate the functions to obtain polynomials that commute with $f(x)$.

The algorithm for determining all permutation polynomials that commute with $f(x)$ is similar to the above algorithm. We only need to add a test for one-to-one function at Step 3.

When the algorithm is implemented, it is not necessary to explicitly construct the matrix in step 3. Because of the circulant structure of the blocks of the matrix, the information needed for constructing the functions can be easily obtained from the size of the blocks. Hence we only need to stored the block sizes (i.e., the lengths of cycles) in a one dimensional array. This will significantly reduce the storage requirement.

In implementing the Lagrange interpolation, we may take advantage of the fact that in the finite field F_q,

$$\prod_{a \in F_q} (x - a) = x^q - x$$

and

$$\prod_{a \in F_q \setminus \{0\}} a = -1.$$

Therefore, the interpolation formula is simplified to

$$g(x) = \sum_{a \in F_q} -f(a) b_a(x),$$

where $b_a(x) = (x^q - x)/(x - a)$.

An inplementation of the algorithm is given by the following pseudocode.

The finite field $GF(p^n)$ is implemented as an n-dimensional vector space over Z_p with the multiplication defined as in $Z_p[x]/\langle m(x) \rangle$ where $m(x)$ is an irreducible polynomial of degree n over Z_p. We apply Berlekamp's factorization algorithm in [1] for polynomials over a finite field to find the irreducible polynomial $m(x)$. The elements of the finite field are denoted by $a_0, a_1, \cdots, a_{q-1}$.

input p and n of $GF(p^n)$; $q := p^n$
construct irreducible polynomial of degree n over Z_p; initialize field operations
input $f(x)$
for $i := 0$ to $q - 1$ do $\sigma[i] = f(a_i)$
find cycle decomposition of σ; k :=number of cycles

sort cycles in increasing order and obtain τ
set μ to the mapping of symbols from σ to τ; comupte μ^{-1}
set l to lengths of cycles
for $i := 0$ **to** k **do**
begin
  index:= 0
  for $j := 0$ **to** k **do**
  begin
    if $l[i]l[j]$ **then**
    begin
      for $m := 0$ **to** $l[j] - 1$ **do begin** $s[i][\text{index}] = a[j] + m$; index := index + 1 **end**
    end
  end
  $s[i][\text{index}] = -1$
end
clear f index to 0; next:=TRUE
while next
begin
  construct function h from f index
  $g := \mu^{-1}h\mu$
  poly:= 0
  for $i := 0$ **to** $q - 1$ **do** poly:= poly $- g[i] * (x^q - x)/(x - a_i)$
  output poly
  index:= 0; next:= FALSE; $i := 0$
  while next=FALSE **and** $i < k$) **do**
    if $s[i]f[i] + 1] \geq 0$ **then begin** f index$[i] :=$ f index$[i] + 1$; next:=TRUE **end**
    else f index$[i] := 0$
  end
end

3. Complexity

In this section we give a complexity analysis of our algorithm and compare it with the complexity of the "brute force" approach. We will only consider the field operations in the algorithms, since they are the most complicated operations and other operations in our algorithm such as the integer comparisions in step 3 are insignificant comparing to field operations.

We assume that the "brute force" approach to the problem is to exame every polynomial $g(x)$ of degree less than q for commutativity with $f(x)$ by evaluating $f(g(x))$ and $g(f(x))$ at every element of the finite field.

Since there are q^q polynomials of degree less than q and there are q elements in the finite field, $4q^{q+1}$ polynomial evaluations are needed in the brute force approach. To count the number of field operations, we assume that Horne's method is used for polynomial evaluations. Evaluating a polynomial of degree m at one point requires $2m$ operations (m

multiplications and m additions). In the worst case when the degree of $f(x)$ is $q-1$, and evaluation of $f(x)$ requires $2(q-1)$ operations. There are $(q-1)q^m$ different polynomials of degree m for $m \geq 0$ and one 0 polynomial. Hence the total number of field operations in the worst case is

$$\sum_{m=0}^{q-1} 2[2(q-1)+2m]q(q-1)q^m = O(q^{q+2}).$$

On average, assume that the degree of $f(x)$ is $(q-1)/2$ and that the average number of points to evaluate for each polynomial is $q/2$. Then the number of field operations is

$$\sum_{m=0}^{q-1} 2[2(q-1)/2+2m](q/2)(q-1)q^m = O(q^{q+2}).$$

The complexity for finding all permutation polynomials that commute with $f(x)$ using the "brute force" approach is the same as the above.

In our algorithm, it requires q evaluations of polynomial $f(x)$ to obtain the permutation σ. A division of $x^q - x$ by $x-a$ requires $3q$ field operations. Hence, each interpolation in the simplified form needs $q(3q)$ operations.

In the worst case, there are q^q functions commuting with $f(x)$. $q!$ of them are permutations. Hence to find all polynomials the total number of field operations needed in our algorithm is

$$2(q-1)^2 + q(3q)q^q = O(q^{q+2}).$$

To find all permutation polynomials, the total number of field operations is

$$2(q-1)^2 + q(3q)q! = O(q^2 \cdot q!).$$

Even in this worst case (which occurs only when $f(x) = x$), our algorithm still offers an improvement. However, the main advantage of our algrithm is that it only generates the polynomials that are actually needed in the result, and that it does not need to process in any way the other polynomials. The "brute force" method, on the other hand, must always test all q^q polynomial functions. Since the number of polynomials that commute with $f(x)$ is usually much less than the number of all polynomials, in the average case our algorithm is significantly more efficient than the "brute force" approach.

In the average case, assume again that the degree of $f(x)$ is $(q-1)/2$. The number of permutations of type $(n_1, n_2, \cdots, n_q)$ is

$$q! \prod_{k=1}^{n} \frac{1}{n_k!(k^{n_k})}.$$

Hence the average number of polynomials of degree less than q that commute with a given permutation polynomial is

$$\sum_{(n_1,n_2,\cdots,n_q)} \prod_k (\sum_{j|k} j n_j)^{n_k} \frac{1}{k^{n_k} n_k!},$$

where the sum is over all partitions of q.

Similarly the average number of permutation polynomials that commute with a given permutation polynomial is

$$\sum_{(n_1,n_2,\cdots,n_q)} \prod_k 1 = p(q),$$

where $p(q)$ is the number of partitions of q. It is well known that (see [3])

$$p(q) = O(e^{\pi\sqrt{\frac{2}{3}\sqrt{q}}}).$$

This is clearly a significant improvement over the "brute force" approach.

4. Examples

In this section we present some examples of using the algorithm in section 2 and the results of the implementation.

Example 1 Consider the polynomial $f(x) = x + \alpha$ over the finite field F_4, where α is a primitive element of F_4. It is easy to verify that $f(x)$ is a permutation polynomial. Applying our algorithm, we obtained the following 16 polynomials that commute with $f(x)$.

$$\begin{aligned}
&\alpha x^2 + \alpha x,\\
&x^2 + (\alpha+1)x + \alpha,\\
&x + 1,\\
&(\alpha+1)x^2 + (\alpha+1),\\
&x^2 + (\alpha+1)x,\\
&\alpha x^2 + \alpha x + \alpha,\\
&(\alpha+1)x^2 + 1,\\
&x + (\alpha+1),\\
&x,\\
&(\alpha+1)x^2 + \alpha,\\
&\alpha x^2 + \alpha x + 1,\\
&x^2 + (\alpha+1)x + (\alpha+1),\\
&(\alpha+1)x^2,\\
&x + \alpha,\\
&x^2 + (\alpha+1)x + 1,\\
&\alpha x^2 + \alpha x + (\alpha+1).
\end{aligned}$$

Example 2 Let $f(x) = x^{15} + 16x^{14} + x^{13} + 16x^{12} + x^{11} + 16x^{10} + x^9 + 16x^8 + x^7 + 16x^6 + x^5 + 16x^4 + x^3 + 16x^2 + 2x$. Then $f(x)$ is a permutation polynomial over F_{17}. Applying our algorithm, we obtain the following 17 polynomials that commute with $f(x)$.

0,

x,

$x^{15}+16x^{14}+x^{13}+16x^{12}+x^{11}+16x^{10}+x^9+16x^8+x^7+16x^6+x^5+16x^4+x^3+16x^2+2x$,

$3x^{15}+12x^{14}+9x^{13}+16x^{11}+3x^{10}+10x^9+15x^8+3x^7+12x^6+9x^5+16x^3+3x^2+11x$,

$6x^{15}+3x^{14}+2x^{13}+4x^{12}+4x^{11}+5x^{10}+4x^9+16x^8+4x^6+16x^5+13x^4+11x^3+x^2$,

$10x^{15}+4x^{14}+15x^{13}+3x^{12}+8x^{11}+6x^{10}+15x^8+4x^7+5x^6+12x^5+12x^4+15x^3+2x^2+13x$,

$15x^{15}+13x^{14}+4x^{13}+7x^{12}+5x^{11}+4x^{10}+10x^9+16x^8+16x^7+13x^6+6x^5+8x^4+x^3+4x^2+3x$,

$4x^{15}+11x^{14}+16x^{13}+3x^{12}+12x^{11}+13x^{10}+7x^9+10x^7+15x^6+11x^5+12x^4+11x^3+12x^2+6x$,

$11x^{15}+13x^{14}+2x^{13}+16x^{12}+6x^{11}+4x^{10}+2x^9+x^8+3x^7+13x^6+8x^5+16x^4+4x^2+11x$,

$2x^{15}+4x^{13}+15x^{11}+11x^7+10x^5+9x^3+9x$,

$11x^{15}+4x^{14}+2x^{13}+x^{12}+6x^{11}+13x^{10}+2x^9+16x^8+3x^7+4x^6+8x^5+x^4+13x^2+11x$,

$4x^{15}+6x^{14}+16x^{13}+14x^{12}+12x^{11}+4x^{10}+7x^9+10x^7+2x^6+11x^5+5x^4+11x^3+5x^2+6x$,

$15x^{15}+4x^{14}+4x^{13}+10x^{12}+5x^{11}+13x^{10}+10x^9+x^8+16x^7+4x^6+6x^5+9x^4+x^3+13x^2+3x$,

$10x^{15}+13x^{14}+15x^{13}+14x^{12}+8x^{11}+11x^{10}+2x^8+4x^7+12x^6+12x^5+5x^4+15x^3+15x^2+13x$,

$6x^{15}+14x^{14}+2x^{13}+13x^{12}+4x^{11}+12x^{10}+4x^9+x^8+13x^6+16x^5+4x^4+11x^3+16x^2$,

$3x^{15}+5x^{14}+9x^{13}+16x^{11}+14x^{10}+10x^9+2x^8+3x^7+5x^6+9x^5+16x^3+14x^2+11x$,

$x^{15}+x^{14}+x^{13}+x^{12}+x^{11}+x^{10}+x^9+x^8+x^7+x^6+x^5+x^4+x^3+x^2+2x$.

Note if the "brute force" method is applied to this example, over 17^{17} field operations are required, which is certainly beyond the capibility of any computer.

Example 3 Over the finite field F_{16}, let $f(x)=\alpha x^6+\alpha x^5+x^4+(\alpha^2+1)x^3+(\alpha+1)x^2+\alpha^2x+1$, where α is a primitive element. The following is list of 15 polynomials that commute with $f(x)$. All of them are permutation polynomials.

x,

$(\alpha^2+\alpha+1)x^6+(\alpha^2+\alpha)x^5+\alpha x^4+(\alpha^2+1)x^2+(\alpha^2+\alpha+1)x$,

$(\alpha^2+\alpha+1)x^6+(\alpha^2+\alpha+1)x^4+\alpha^2x^3+x^2+\alpha^2x$,

$(\alpha^2+1)x^6+\alpha^2x^5+(\alpha+1)x^4+(\alpha^2)x^3+(\alpha^2+\alpha)x^2+\alpha x+1$,

$\alpha x^6+\alpha x^5+x^4+(\alpha^2+1)x^3+(\alpha+1)x^2+\alpha^2x+1$,

$\alpha x^6+\alpha^2x^5+\alpha^2x^4+x^3+(\alpha^2+\alpha+1)x^2+(\alpha^2+\alpha+1)x+1$,

$\alpha x^6+(\alpha+1)x^4+(\alpha+1)x^3+\alpha^2x^2+(\alpha^2+\alpha+1)x+\alpha$,

$(\alpha^2+1)x^6+(\alpha^2+\alpha)x^5+x^4+(\alpha+1)x^3+x^2+x+\alpha$,

$(\alpha^2+1)x^6+\alpha^2x^4+(\alpha^2+\alpha+1)x^3+(\alpha^2+1)x^2+\alpha x+\alpha$,

$\alpha x^6+(\alpha^2+1)x^5+(\alpha+1)x^4+(\alpha^2+1)x^2+(\alpha^2+\alpha)x+(\alpha+1)$,

$(\alpha^2+1)x^6+(\alpha+1)x^5+x^4+(\alpha+1)$,

$(\alpha^2+1)x^6+(\alpha^2+1)x^5+\alpha^2x^4+\alpha^2x^3+\alpha^2x^2+(\alpha+1)x+(\alpha+1)$,

$(\alpha^2+1)x^6+(\alpha^2+\alpha+1)x^5+(\alpha^2+\alpha)x^4+\alpha x^3+(\alpha+1)x^2+x+\alpha^2$,

$$\alpha x^6 + x^5 + \alpha^2 x^4 + \alpha x^3 + (\alpha^2 + \alpha)x^2 + (\alpha^2 + \alpha + 1)x + \alpha^2,$$
$$\alpha x^6 + (\alpha^2 + \alpha + 1)x^5 + x^4 + (\alpha^2 + \alpha)x^3 + \alpha x^2 + \alpha^2 x + \alpha^2.$$

References:

[1] BERLEKAMP E R. *Factoring polynomials over finite field* [J]. Bell System Tech. J., 1967, **46**: 1853–1859.

[2] CHAO C Y. *Polynomials over finite fields which commute with a permutation polynomial* [J]. J. of Algebra, 1994, **163**(2): 295–311.

[3] HUA L K. *Introduction to Number Theory* [M]. Springer-Verlag, Berlin, 1982.

[4] MULLEN G L. *Polynomials over finite fields which commute with linear permutations* [J]. Proc. Amer. Math. Soc., 1982, **84**: 315–317.

[5] WELLS C. *Polynomials over finite fields which commute with translation* [J]. Proc. Amer. Math. Soc., 1974, **46**: 347–350.