A Hybrid Ben-Or/Tiwari-Zippel Sparse Interpolation Algorithm

Erich Kaltofen and Wen-shin Lee
North Carolina State University
www.math.ncsu.edu/~kaltofen
www.math.ncsu.edu/~wlee1

East Coast Computer Algebra Day
April 24, 1999

Input:

A black box that evaluates $f(x_1, x_2, \ldots, x_n) \in K[x_1, x_2, \ldots, x_n]$, where K is a field of a prime characteristic q, and the characteristic q.
An upper bound of the total degree of f, $\mathrm{degbd}(f)$, and a list of variables of f, $[x_1, x_2, \ldots, x_n]$.
The size of the random number generator.

Output:

With probabilistically correct, $\tilde{f}(x_1, x_2, \ldots, x_n) \in K[x_1, x_2, \ldots, x_n]$ that evaluates the input black box.
An error message will be returned if the input is invalid or the algorithm fails.

Step 1: Interpolate the Polynomial with respect to the Introduced Homogenizing Variable

Introduce a new variable x_0, which we call homogenizing variable, and consider $f(x_0, x_1, x_2, ..., x_n) = f(x_0 x_1, x_0 x_2, ..., x_0 x_n)$.

Interpolate $f(x_0, x_1, x_2, ..., x_n)$ with respect to x_0, and $f_0(x_0)$ will be the resulting polynomial with coefficients in $K[x_1, x_2, ..., x_n]$.

Generate $a_0, a_1, a_2, ..., a_n$ from the random number generator. We call $(a_0, a_1, a_2, ..., a_n)$ the anchor point.

Step 1.a:

Define $a_{0,j} = (a_0^j, a_1, a_2, \ldots, a_n)$.
For $i = 1, 2, \ldots, (\operatorname{degbd}(f) + 1)$ Do **Step 1.a.i**, **Step 1.a.ii**, and **Step 1.a.iii**

Step 1.a.i:

Use Newton's algorithm to interpolate $f_0(x_0)$ at $a_{0,1}, a_{0,2}, \ldots, a_{0,i}$, and this interpolation polynomial is called f_0^i.

If $f_0^i = f_0^{i-1}$, then **Step 1.a.i** is complete, and $f_0^i = f_0$.

Step 1.a.ii:

Implement Berlekamp-Massey algorithm on $a_{0,1}, a_{0,2}, \ldots, a_{0,i}$.

If we get zero discrepancy, then the generating polynomial is found for $a_{0,1}, a_{0,2}, \ldots, a_{0,i}$. The terms of f_0 can be determined by factoring this generating polynomial.

By Ben-Or/Tiwari algorithm, the corresponding coefficients can be obtained. And thus f_0 may be found.

If f_0 is found successfully, **Step 1.a.ii** is complete. This usually happens when f_0 only has a few terms.

Step 1.a.iii:

If either **Step 1.a.i** or **Step 1.a.ii** is complete, **Step 1.a** is finished.

If neither **Step 1.a.i** nor **Step 1.a.ii** is complete and the loop variable $i = \mathrm{degbd}(f) + 1$, return an error message.

Check:

Whether $\deg(f_0) \leq \mathrm{degbd}(f)$.
If not, return an error message.

Step 2: Interpolate Next Variable x_i

We now have a polynomial $f_{i-1}(x_0, x_1, \ldots, x_{i-1})$, which has been fully interpolated in $x_0, x_1, \ldots, x_{i-1}$.

$p_0, p_1, ..., p_{i-1}$ are generated from the random number generator. Define $p_{i,j} = (p_0, p1, ..., p_{i-1}, a_i^j, a_{i+1}, a_{i+2}, ..., a_n)$.

Step 2.a:

Consider each coefficient in f_{i-1}, when f_{i-1} is evaluated at $p_{i,1}, p_{i,2}, ..., p_{i,j}$, this coefficient is a polynomial in x_i. We now try to interpolate all those coefficient polynomials in f_{i-1}.

An upper bound of the degree of a such coefficient polynomial, $\text{degbd}(C_{e_0,e_1,...,e_{i-1}})$ can be obtained by $[\deg(f_0) -$ (the degree of the corresponding monomial)$]$.

$f_i = \Sigma_{(e_0,e_1,...,e_{i-1})\in J}\, C_{e_0,e_1,...,e_{i-1}} x_0^{e_0} x_1^{e_1} \cdots x_{i-1}^{e_{i-1}}$

For every $C_{e_0,e_1,\ldots,e_{i-1}}$ with $(e_0, e_1, \ldots, e_{i-1}) \in J$ Do

For $j = 1, 2, \ldots, +\operatorname{degbd}(C_{e_0,e_1,\ldots,e_{i-1}}) + 1$ Do **Step 2.a.i**, **Step 2.a.ii**, and **Step 2.a.iii**

Step 2.a.i:

Use Newton's algorithm to interpolate $C_{e_0,e_1,\ldots,e_{i-1}}(x_i)$ at $p_{i,1}, p_{i,2}, \ldots, p_{i,j}$, and this interpolation polynomial is called $C^j_{e_0,e_1,\ldots,e_{i-1}}$.

If $C^j_{e_0,e_1,\ldots,e_{i-1}} = C^{j-1}_{e_0,e_1,\ldots,e_{i-1}}$, then **Step 2.a.i** is complete, and $C^j_{e_0,e_1,\ldots,e_{i-1}} = C_{e_0,e_1,\ldots,e_{i-1}}$.

Step 2.a.ii:

Implement Berlekamp-Massey algorithm on $p_{i,1}, p_{i,2}, \ldots, p_{i,j}$.

If we get zero discrepancy, then the generating polynomial is found for $p_{i,1}, p_{i,2}, \ldots, p_{i,j}$. The terms of $C_{e_0,e_1,\ldots,e_{i-1}}$ can be determined by factoring this generating polynomial.

By Ben-Or/Tiwari algorithm, the corresponding coefficients can be obtained. And thus $C_{e_0,e_1,\ldots,e_{i-1}}$ may be found.

If $C_{e_0,e_1,\ldots,e_{i-1}}$ is found successfully, **Step 2.a.ii** is complete. This usually happens when $C_{e_0,e_1,\ldots,e_{i-1}}$ only has a few terms.

Step 2.a.iii:

If either **Step 2.a.i** or **Step 2.a.ii** is complete, **Step 2.a** for $C_{e_0,e_1,\dots,e_{i-1}}$ is finished.

If neither **Step 2.a.i** nor **Step 2.a.ii** is complete for $C_{e_0,e_1,\dots,e_{i-1}}$ and the loop variable $j = \mathrm{degbd}(C_{e_0,e_1,\dots,e_{i-1}}) + 1$, return an error message.

Check:

Whether $\deg(C_{e_0,e_1,\dots,e_{i-1}}) \leq \mathrm{degbd}(C_{e_0,e_1,\dots,e_{i-1}})$.
If not, return an error message.

Step 2.b:

Use Zippel's algorithm, we prune all the monomials in $x_0, x_1, \ldots, x_{i-1}, x_i$ whose coefficients are zero. With probabilistically correct, the resulting polynomial f_i is fully interpolated in $x_0, x_1, \ldots, x_{i-1}, x_i$.

Step 2.c:

If f_n is determined, **Step 2** is finished.

Step 3: Recover $\tilde{f}$ from f_n

Remove all x_0 in the algebraic expression of f_n will get $\tilde{f}$.

Implementation:

Our new hybrid algorithm is implemented in a Maple program.