Computing polynomials using black box representation

Garrett Paluck. Department of Mathematics, Simon Fraser University, British Columbia.

In science, computing, and engineering, a black box is a device, system or object where the inputs and outputs are observable, but the functionality of the black box is not. Here, a black box is a computer program (procedure) that outputs a value, but we cannot view the code.

Algorithm Descriptions

Let $f \in \mathbb{Z}[x_1, x_2, ..., x_n]$. In our implementation of the black box, we create a procedure which simulates the black box. The black box receives integer input for n variables and a prime p, and outputs an integer in $\mathbb{Z}_p$. See figure 1 below.

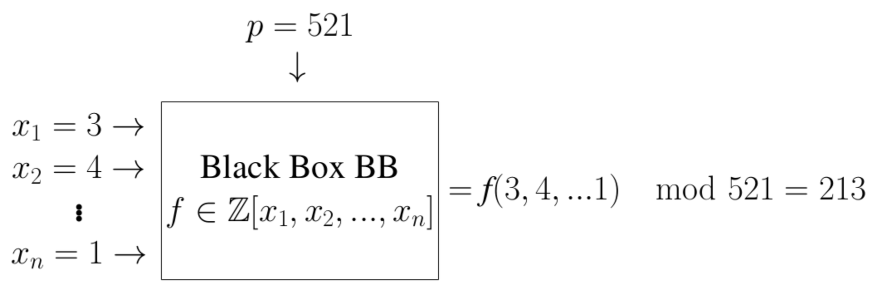

Figure 1:BB Concept

Using this evaluation procedure, we wish to successfully implement the four following functions:

- **isBBZero** - Checks whether f is the 0 polynomial.
- **degBB** - Outputs the total degree of the polynomial f, or the maximum degree of one of the n variables.
- **suppBB** - Outputs the support, i.e. the monomials of f, $\{M_1, M_2, ..., M_t\}$.
- **sintBB** - Outputs the polynomial f, $f = \sum_{i=1}^{t} a_i M_i$.

We require several variables for algorithm functionality and analysis. Let BB represent the unknown polynomial f, D be a degree bound of f, T be a term bound on f, and H be a height bound of f.

The BB procedure consists of 2 major operations: evaluating the n variables in f and preforming modular division using a prime p. For analysis, will count the number of calls to the black box BB.

Algorithm Analysis

isBBZero

This procedure determines if f is the zero polynomial. We accomplish this by evaluating f at $\alpha \in S^n$ where S=$[0, p-1]$ and p is a prime. We then evaluate BB$(\alpha) \mod p$ and verify whether the results are 0 for multiple primes. The pseudocode for the algorithm is below.

Input: BB,D, H,ϵ $D \geq \deg f$, $H \geq ||f||$
Output: true/false
$n \leftarrow$ # variables
error $\leftarrow$ 1
$M \leftarrow 1$
while error $> \epsilon$ **do**
  $j \leftarrow 1$
  $p_j \leftarrow$ random prime $\in [10^{10}, 2 \cdot 10^{10}]$
  pick $\alpha \in \mathbb{Z}_p^n$ at random
  eval$_j \leftarrow$BB$(\alpha) \mod p$ $\leftarrow O$(BB)
  while $M \leq 2H$ **do**
    $j \leftarrow j + 1$
    $p_j \leftarrow$ random prime $\in [10^{10}, 2 \cdot 10^{10}]$
    $M \leftarrow M \cdot p_j$
    pick $\alpha \in \mathbb{Z}_p^n$ at random
    eval$_j \leftarrow$BB$(\alpha) \mod p$ $\leftarrow O$(BB)
  end
  eval$_1 \leftarrow$ Use CRT using evaluation points and corresponding primes $\leftarrow O(\frac{\log H}{\log(\frac{D}{p})})$
  if eval$_1 \neq 0$ **then**
    Return false
  end
  error $\leftarrow$ error$\cdot(D/p_1 p_2 ... p_j)$
end
Return true

Algorithm 1: isBBZero Algorithm

Consider when a nonzero polynomial would evaluate to 0. Firstly, when a prime p divides every co-efficient of the determinant polynomial. To avoid this, if $p < 2H$, then we can use the Chinese Remainder Theorem with multiple primes. Secondly, we must consider when BB$(\alpha_1, \alpha_2, ...\alpha_n) \mod p = 0$. By the Schwartz-Zippel Lemma, Prob[BB$(\alpha_1, \alpha_2, ...\alpha_n) \mod p = 0] \leq \frac{\deg f}{|S|}$= $\frac{D}{|\mathbb{Z}_p|}$ = $\frac{D}{p}$. By using multiple primes we can reduce the probability of selecting bad evaluation points to within an error bound ϵ.

The algorithm is restricted by procedure BB. Assuming that |log H| < |log ϵ|, there will be approximately $\log \epsilon / \log(\frac{D}{p})$ evaluation points. So, this algorithm has a time complexity of $(\frac{\log \epsilon}{\log(\frac{D}{p})}) O(BB)$.

degBB

We can use a similar method to find the total degree d of f. Let $\alpha \in S^n$, S = $[0, p-1]$. We evaluate BB at $D+1$ unique evaluation points, then interpolate a univariate polynomial of degree d.

Input: BB,D $D \geq \deg f$
Output: total degree d of f
$n \leftarrow$ # variables
$p \leftarrow$ random prime $\in [10^{10}, 2 \cdot 10^{10}]$
pick $\alpha \in \mathbb{Z}_p^n$ at random
C $\leftarrow$ BB$(i\alpha) \mod p, i = 0, 1, 2, ..., D$ $\leftarrow (D+1)O$(BB)
E $\leftarrow$ Interpolate C with evaluation points 0,1,2,...,D $\leftarrow O((D+1)^2)$
Return $d \leftarrow$ max degree of E

Algorithm 2: degBB Algorithm

For the probabilistic analysis of procedure **degBB**, consider Theorem 1.

Theorem 1. *Let $f \in \mathbb{Z}[x_1, x_2, ..., x_n]$ with total degree d and $\alpha \in S^n$ at random where $S \subset \mathbb{Z}$. If $g(y) = f(\alpha_1 y, \alpha_2 y ..., \alpha_n y)$ and S=*$[0, p-1]$*, then* Prob[$\deg g(y) < d$] $\leq \frac{D}{p}$.

Proof. $g(y) = f(\alpha_1 y, \alpha_2 y ..., \alpha_n y)$

$= f_d(\alpha_1 y, \alpha_2 y ..., \alpha_n y) + \sum_{i=0}^{d-1} f_i(\alpha_1 y, \alpha_2 y ..., \alpha_n y)$

We are examining the Prob[deg $g(y) < d$], so we need only observe the sum of monomial terms of degree d.

$f_d(\alpha_1 y, \alpha_2 y ..., \alpha_n y) = \sum_{j=1}^{t} a_j (\alpha_1 y)^{e_{1j}} (\alpha_2 y)^{e_{2j}} ... (\alpha_n y)^{e_{nj}}$

$= y^d \sum_{j=1}^{t} a_j (\alpha_1)^{e_{1j}} (\alpha_2)^{e_{2j}} ... (\alpha_n)^{e_{nj}} = y^d f_d(\alpha_1, \alpha_2 ..., \alpha_n)$

The degree of $g(y)$ is less than d, when $y^d f_d(\alpha_1, \alpha_2 ..., \alpha_n) = 0$. As we are in a field, $y^d f_d(\alpha_1, \alpha_2 ..., \alpha_n) = 0$ when either $y^d = 0$ or $f_d(\alpha_1, \alpha_2 ..., \alpha_n) = 0$. As we are in the field $\mathbb{Z}_p$, $y^d \mod p \neq 0$ unless $y = 0$. By the Schwartz-Zippel Lemma, Prob[$f_d(\alpha_1, \alpha_2 ..., \alpha_n) = 0$] $\leq \frac{\deg f}{|S|}$.

So, Prob[deg $g(y) < d$] = Prob[$f_d(\alpha_1, \alpha_2 ..., \alpha_n) = 0$]
$\leq \frac{\deg f}{|S|} = \frac{d}{\mathbb{Z}_p} = \frac{d}{p} \leq \frac{D}{p}$

□

This algorithm has a probability to fail when the interpolated polynomial has a degree less than the total degree. Using Theorem 1, Prob[deg $g(t) < d$] $\leq \frac{\deg f}{|S|} = \frac{d}{p} < \frac{d}{10^{10}}$. As d will be much smaller than 10^{10}, we can say with high probability that the algorithm will output the total degree successfully. The algorithm's run time is dominated by having to call BB $(D+1)$ times, so that algorithm has a running time of D$\cdot O$(BB).

suppBB

This procedure will find the support of f, $\{M_1, M_2, ..., M_t\}$. We adapt Ben-Or/Tiwari's Interpolation algorithm[1] to calculate the support and the Berlekamp-Massey algorithm for finding the minimum polynomial in a field[3].

Input BB,D, T $D \geq \deg f$, $T \geq \# f$
Output A
$n \leftarrow$ # variables
$q \leftarrow$ prime such that $q > p_n^D$
$V_i \leftarrow$ BB$(\alpha) \mod p, i = 0..2T-1$ $\leftarrow 2T \cdot O$(**BB**)
$\lambda \leftarrow$ Berlekamp-Massey Algorithm$(V, q, z) \in \mathbb{Z}_q[z]$ $\leftarrow O(T^2)$
$R \leftarrow$ Roots(λ) $\leftarrow O(T^2 \log q)$
$t \leftarrow$ # roots of R
for *i=1 to t* **do**
  factor R_i over $\mathbb{Z}$, $R_i \leftarrow \sum_{j=1}^{n} p_j^{\alpha_j}$ $\leftarrow O(TD)$
  $M_i \leftarrow \sum_{j=1}^{n} x_j^{\alpha_j}$
end
A $\leftarrow [M_1, M_2, ..., M_t]$
Return A

Algorithm 3: suppBB Algorithm

This is a deterministic algorithm, so it will always output the support successfully. The number of arithmetic operations is dominated from having to call BB $2T$ times and having to find the roots of a polynomial R using Rabin's factoring algorithm. The algorithm requires $2T \cdot O$(BB) arithmetic operations.

sintBB

This procedure will find the coefficients and monomials of f. First, it uses **suppBB** to calculate the monomial terms, then uses Zippel's Algorithm[4] for solving transposed Vandermonde systems in $O(T^2)$ to find the coefficients.

Input BB,D, T, H $D \geq \deg f$, $T \geq \# f$, $H \geq ||f||$
Output k
$M \leftarrow$ **suppBB**(BB,D, T) $\leftarrow O$(**suppBB**)
$n \leftarrow$ # variables, $t \leftarrow$ # terms in M
for *i=1 to t* **do**
  $M_i \leftarrow M_i(2, 3, 5, ..., p_n)$
end
while *true* **do**
  $q_m \leftarrow$ random prime $\in [2^{62}, 2^{63}]$, $q > p_n^D$
  $v_i \leftarrow$BB$(2^i, 3^i, ..., p_n^i) \mod p, i = 0..t-1$ $\leftarrow T \cdot O$(BB)
  $g \leftarrow (z - M_1)(z - M_2)...(z - M_t)$
  $s_i \leftarrow g/(z - M_i) \mod q, i = 1..t$
  $R_i \leftarrow \frac{s_i(z)}{s_i(M_i)} \mod q, i = 1..t$
  $V_{i,j}^{-1} \leftarrow$ coeff$(R_i, z, j-1)$
  $A \leftarrow V^{-1} \cdot v$ $\leftarrow O(T^2)$
  $k \leftarrow \sum_{i=1}^{t} A_i M_i \in \mathbb{Z}_q[x_1, x_2, ..., x_n]$
  Preform the Chinese Remainder Theorem mod q_m on k until the product of primes exceeds $2H$
  Return k
end

Algorithm 4: sintBB Algorithm

This is a deterministic algorithm, so it will always output the support successfully. The number of arithmetic operations is dominated from having to call BB, so this algorithm has the same run time: $T \cdot O$(BB).

Example

We have implemented every algorithm described in Maple. For our implementation, we have created an $m \times m$ matrix of polynomials, and BB evaluates the matrix at $\alpha \in \mathbb{Z}_p^n$, then takes the determinant of the matrix modular some prime p. This process takes $O(m^3 + m^2 TD)$ Please consider the following matrix:

$$\text{BB} = \det\left(\begin{bmatrix} -69\, x_1{}^3 x_2{}^7 & 62\, x_1{}^5 x_2{}^3 \\ -68\, x_1{}^4 x_2{}^{12} x_3{}^2 & 84\, x_1{}^2 x_2{}^{10} x_3{}^6 \end{bmatrix}\right)$$

Let D = 30, T = 10, H = 10000, and $\epsilon = 10^{-50}$
isBBZero(B, D, H,ϵ) = false
degBBZero(B, D) = 28
suppBB(B, D, T) = $[x_1^5 x_2^{17} x_3^6, x_1^9 x_2^{15} x_3^2]$
sintBB(B, D, T, H) = $-5796\, x_1^5 x_2^{17} x_3^6 + 4216\, x_1^9 x_2^{15} x_3^2$

References

[1] Michael Ben-Or, and Prasoon Tiwari. A Deterministic Algorithm for Interpolating Sparse Multivariate Polynomials by Ben-Or and Tiwari. *Proc. of STOC 1988, 301–309, ACM Press, 1988*

[2] Michael Monagan, and Jiaxiong Hu. A Fast Parallel Sparse Polynomial GCD Algorithm. *Proc. of ISSAC 2016, 271–278, ACM Press, 2016*

[3] Massey, J.L. "Shift-register synthesis and BCH decoding." *IEEE Transactions on Information Theory* **15** *(1969) 122–127.*

[4] Zippel, Richard. "Interpolating Polynomials from their values." *J. Symbolic Computation* **9** *(1990) 375–403.*