Counting Solutions of Polynomial Equations

Kirill Levchenko and Yi-Kai Liu

In "Improved Range-Summable Random Variable Construction Algorithms" [1] it was claimed that a range sum of a Reed-Muller hash function can be computed in polynomial time; however the proof was incorrect. In fact, only second-order Reed-Muller hash functions are known to be efficiently range-summable; the problem of range-summing Reed-Muller hash functions of order three and above is essentially #**P**-complete, in the sense that the number of solutions of the #**P** problem is a linear function of the range range sum, where the function is determined by the reduction. The remainder of this note establishes this fact. It is worth noting that this sort of result has been known for at least ten years, see, e.g., [2].

Recall that an order-r Reed-Muller hash function from $\{0,1\}^n$ to $\{0,1\}$ is an n-variable degree-r polynomial over GF(2). Computing the range-sum of the hash function is thus equivalent to counting the number of times the polynomial evaluates to zero. We show that we can reduce #SAT to the problem of counting the number of roots of a degree-3 polynomial. The number of solutions to the polynomial will be a linear function of the number of solutions of the 3CNF formula, where the function will be determined by the reduction.

Claim 1. *Let ϕ be an n-variable 3CNF formula. Then ϕ can be transformed, in polynomial time, to a polynomial π such that*

$$|\{\vec{x} : \phi(x) = 1\}| = |\{\vec{x} : \pi(x) = 1\}|.$$

Proof. Recall that for a pair of Boolean variables x and y, we have

$$\begin{aligned} \bar{x} &= 1 + x \\ x \vee y &= x + y + xy \\ x \wedge y &= x \cdot y \end{aligned}$$

where "$+$" and "$\cdot$" are over GF(2). Now replace each clause $\mathcal{C}_i$ of ϕ with the equivalent polynomial $c_i(\vec{x})$. For example,

$$\begin{aligned} x_1 \vee \bar{x}_2 \vee x_3 &= \big(x_1 + (1 + x_2) + x_1(1 + x_2)\big) \vee x_3 \\ &= (1 + x_2 + x_1x_2) \vee x_3 \\ &= 1 + x_2 + x_1x_2 + x_3 + (1 + x_2 + x_1x_2) \cdot x_3 \\ &= 1 + x_2 + x_1x_2 + x_2x_3 + x_1x_2x_2. \end{aligned}$$

Note that each $c_i(\vec{x})$ has degree at most 3 because each clause $\mathcal{C}_i$ has at most three literals. Taking the product of the $c_i(\vec{x})$ gives the desired polynomial

$$\pi(\vec{x}) = \prod_i c_i(\vec{x}).$$

□

The resulting polynomial $\pi(\vec{x})$ is, in general, of arbitrary degree. We transform it to a degree-3 polynomial p on $n+\ell$ variables, such that the number of solutions to p is related to the number of solutions to ϕ. To do this, we first transform π into a system of polynomial equations where each equation has degree at most 2.

Claim 2. *An n-variable polynomial π given as a product of m degree-3 polynomials, can be transformed, in polynomial time, into a system of at most $\ell = 3m+1$ polynomial equations, each of degree at most 2, such that π and the resulting system of equations have the same number of solutions.*

Proof. We start with the system of equations containing only $\pi(\vec{x}) = 1$. We then reduce the degree of π at the cost of introducing additional equations into the system.

Each clause polynomial $c_i(\vec{x})$ contains at most one term of degree 3, say $x_1x_2x_3$. We replace it with x_1y_{23} and introduce constraint equation $y_{23} = x_2x_3$. Thus, each clause polynomial $c_i(\vec{x})$ may be transformed into a degree-2 polynomial $c'_i(\vec{x})$ at the cost of introducing m equations into the system.

Next, replace each clause $c'_i(\vec{x})$ with a single variable z_i and introduce k equations $z_i = c'_i(\vec{x})$ to the system, so that we now have $\pi(\vec{x}) = \prod_i z_i$. Finally, introduce a third set of equations given by $z'_1 = z_1$ and $z'_i = z'_{i-1}z_i$ for $i > 1$. The initial equation $\pi(\vec{x}) = 1$ thus becomes $z'_m = 1$.

By construction, each assignment to $\vec{x}$ forces a unique assignment to the variables y_i, z_i, and z'_i. Furthermore, each equation introduced into the system has degree at most 2, and the initial equation is replaced with $z'_m = 1$, and each of the three steps introduced at most m equations. □

It remains to turn the system of ℓ degree-2 polynomials into a single polynomial whose solutions are related to solutions of the system.

Claim 3. *Consider an n-variable system of ℓ degree-2 polynomial equations $q_i(\vec{x}) = 0$. The system can be transformed into a single degree-3 polynomial on $n+\ell$ variables such that if the number of satisfying assignments the system of equations is a and then number of roots of the polynomial is b, then $a = 2^{1-\ell}b - 2^n$.*

Proof. Define

$$s_i(\vec{x}) = \frac{1}{2}\sum_{w\in\mathrm{GF}(2)}(-1)^{w\cdot q_i(\vec{x})}.$$

Note that $s_i(\vec{x})$ is 1 if an assignment to $\vec{x}$ satisfies $q_i(\vec{x}) = 0$ and 0 if it does not. It follows that $S(\vec{x}) = \prod_i s_i(\vec{x})$ is 1 if an assignment to $\vec{x}$ satisfies all the equations, and 0 if it does not. Thus, we're after the sum of $S(\vec{x})$ over all possible assignments to $\vec{x}$. Now

$$\begin{aligned}
\sum_{\vec{x}} S(\vec{x}) &= \sum_{\vec{x}} \prod_{i=1}^{\ell} s_i(\vec{x}) \\
&= \sum_{\vec{x}} \prod_{i=1}^{\ell} \frac{1}{2} \sum_{w_i} (-1)^{w \cdot q_i(\vec{x})} \\
&= \sum_{\vec{x}} \frac{1}{2^\ell} \sum_{\vec{w}} \prod_{i=1}^{\ell} (-1)^{w \cdot q_i(\vec{x})} \\
&= \frac{1}{2^\ell} \sum_{\vec{x},\vec{w}} \prod_{i=1}^{\ell} (-1)^{w \cdot q_i(\vec{x})} \\
&= \frac{1}{2^\ell} \sum_{\vec{x},\vec{w}} (-1)^{p(\vec{x},\vec{w})} \\
&= \frac{1}{2^\ell} \left[|\{(\vec{x},\vec{w}) \,:\, p(\vec{x},\vec{w}) = 0\}| - |\{(\vec{x},\vec{w}) \,:\, p(\vec{x},\vec{w}) = 1\}| \right] \\
&= \frac{1}{2^\ell} \left[2|\{(\vec{x},\vec{w}) \,:\, p(\vec{x},\vec{w}) = 0\}| - 2^{n+\ell} \right]
\end{aligned}$$

where p is a degree-3 polynomial on $n+\ell$ variables. So $\sum S(\vec{x})$ is

$$2^{1-\ell} |\{(\vec{x},\vec{w}) \,:\, p(\vec{x},\vec{w}) = 0\}| - 2^n.$$

□

References

[1] A. R. Calderbank, A. Gilbert, K. Levchenko, and S. Muthukrishnan. "Improved Range-Summable Random Variable Construction Algorithms." *Proc. of the 16th Annual ACM-SIAM Symposium on Discrete Algorithms*. pp. 840–849 (2005).

[2] A. Ehrenfeucht and M. Karpinski. "The Computational Complexity of (XOR,AND)-Counting Problems." *Technical Report TR-90-031*, International Computer Science Institute, Berkeley, 1990.