Computational Number Theory and Algebra June 6, 2012

Lecture 14

Lecturers: Markus Bläser, Chandan Saha *Scribe: Chandan Saha*

In the past two lectures, we have introduced Hensel lifting and shown how this tool can be applied to prove that factoring of bivariate polynomials over a field reduces (in polynomial time) to univariate polynomial factoring over the same field. A similar reduction holds from n-variate polynomial factoring to univariate polynomial factoring for any n (but, of course, the reduction time grows exponentially in n). Algorithmically speaking, Hensel lifting is an efficient procedure - it simply involves multiplication of elements from the underlying (polynomial) ring (refer to the proofs of lemma 3 & 4 in Lecture 12). But, nevertheless, it would be nice if we can somehow shift the complexity of 'Hensel lifting involving n variables' from the algorithm to the analysis, and in the process make the algorithm somewhat 'simpler'. This would be the topic of discussion today. The primary tool that would help us achieve this is Hilbert's irreducibility theorem, which was mentioned in the last class. (Refer to Madhu Sudan's course lecture note 9 and observe how Hensel lifting is used in the proof of Hilbert's irreducibility theorem.) The idea is to use this theorem to design an algorithm for factoring n-variate polynomials for any n, where the algorithm uses 'Hensel lifting involving *only* 3 variables'. This somewhat 'simplification' of the factoring algorithm comes at a price - randomization! This is because, Hilbert's irreducibility theorem itself involves randomization (see the statement of the theorem in the previous lecture note).

The overall strategy is to factor the polynomials in a 'black-box fashion' using Hilbert's irreducibility theorem and then apply polynomial interpolation to retrieve the factors. These are the topics that we will discuss today:

- Black-box polynomial factoring, and
- Sparse polynomial interpolation.

A few words about the black-box model- Before we proceed to the main part of our discussion, let us say a few words about the 'black-box' model and why we study it. One motivation behind studying this model comes from the arithmetic circuit model of computation (also known as *straight line programs*). An arithmetic circuit is like a boolean circuit with addition and multiplication gates replacing 'or' and 'and' gates, and the inputs are variables. Naturally, an arithmetic circuit computes a multivariate polynomial. Arithmetic circuits provide a succinct way of representing polynomials - for instance, the polynomial $f = (x_1+1)(x_2+1)\dots(x_n+1)$ has 2^n monomials, but an arithmetic circuit can 'compute' this polynomial using only $n+1$ gates. This means that the polynomial f can be evaluated at a point $(a_1, \dots, a_n)$ using only $O(n)$ addition and multiplication operations which is far fewer than the total number of monomials in f. This is where the 'black-box' model enters the scene: We say that a polynomial f is computed by a 'black-box' if f is *hidden* inside a black box (which is probably hiding an arithmetic circuit computing the polynomial), and given any point $(a_1, \dots, a_n)$ the black-box returns the value $f(a_1, \dots, a_n)$ very efficiently. This means, any algorithm is only allowed to query evaluations of f at certain points to the black-box computing f. The black-box model of computation has profound connections with some of the fundamental problems of complexity theory, including an algebraic analogue of the famous $\mathsf{P} \stackrel{?}{=} \mathsf{NP}$ question.

In the context of polynomial factoring, it is conceivable that there are problems where we are interested in certain evaluations of the factors and not the factors themselves. In such cases, black-box factorization is more efficient than n-variate Hensel lifting. Moreover, the 'black-box' model is interesting for another efficiency reason. It might happen that an n-variate polynomial f, whose total degree is bounded by d, has factors whose sparsity [1] is much less than d^n. If so, then the output size would be much less than d^n. However, it might happen than Hensel lifting takes $O(d^n)$ time to compute the factors. Ideally, we would like to output the factors of f in time polynomial in d, n and m, where m is an upper bound on the sparsity

[1] the number of monomials with nonzero coefficients in a polynomial is called its sparsity.

of the factors of f and m is potentially much less than d^n. This leads us to the topic of sparse polynomial interpolation.

You may find the discussion on black-box factorization of multivariate polynomials in section 2 of lecture 9 from Madhu Sudan's course notes. We will just write about sparse polynomial interpolation in this note.

1 Sparse polynomial interpolation

Suppose an n-variate polynomial $f(x_1, \ldots, x_n) \in \mathbb{F}[x_1, \ldots, x_n]$ is given as a black-box with the promise that the sparsity of f is upper bounded by m and its (total) degree is bounded by d. We want to find f, which means we want to express f in the sum of monomials form (also known as the sparse representation of f), in time polynomial in n, d and m by making queries to the black-box at points in $\mathbb{F}^n$ (or, $\mathbb{K}^n$ where $\mathbb{K} \supset \mathbb{F}$ is a 'small' extension field of $\mathbb{F}$). This problem is known as the sparse polynomial interpolation problem. The algorithm we will discuss for this problem is due to Klivans and Spielman [KS01] - it works over any underlying field $\mathbb{F}$.

Let $f = \sum_{i=1}^{m} c_i x_1^{d_{i1}} x_2^{d_{i2}} \ldots x_n^{d_{in}}$, where $c_1, \ldots, c_m$ are the coefficients of the m monomials of f. Let p be a prime whose value will be fixed later. Consider replacing x_i by $x^{t^{i-1} \bmod p}$ in f, where the value of t will also be fixed later.

$$\begin{aligned} f(x, x^{t \bmod p}, x^{t^2 \bmod p}, \ldots, x^{t^{n-1} \bmod p}) &= \sum_{i=1}^{m} c_i x^{d_{i1} + d_{i2}(t \bmod p) + \ldots + d_{in}(t^{n-1} \bmod p)} \\ &= \sum_{i=1}^{m} c_i x^{e_i(t)}, \end{aligned}$$

where $e_i(t) = d_{i1} + d_{i2}(t \bmod p) + \ldots + d_{in}(t^{n-1} \bmod p)$. Define $p_i(t)$ as,

$$p_i(t) \stackrel{\text{def}}{=} (d_{i1} + d_{i2}t + \ldots + d_{in}t^{n-1}) \mod p.$$

For the moment, pretend that t is a variable. Then, $p_i(t)$ can be naturally treated as a polynomial in t over $\mathbb{F}_p$ with $\deg p_i(t) < n$. Now suppose $p > d$, then $p_i(t) \neq p_j(t)$ for any $i \neq j$ (as distinct indices i and j correspond to distinct monomials and hence $(d_{i1}, \ldots, d_{in}) \neq (d_{j1}, \ldots, d_{jn})$). This means that the polynomial $q_{ij}(t) \stackrel{\text{def}}{=} p_i(t) - p_j(t)$ is a non-zero polynomial over $\mathbb{F}_p$ (for $i \neq j$) with degree bounded by n. Moreover, every (unordered) pair (i, j), with $i \neq j$, defines a non-zero polynomial $q_{ij}(t)$. Since, there are $\binom{m}{2}$ pairs (i, j), there are $\binom{m}{2}$ such polynomials $q_{ij}(t)$. Therefore, if $p > \binom{m}{2} \cdot n$ then there is an $\alpha \in \mathbb{F}_p$ such that none of the $q_{ij}(\alpha)$ is zero, implying that $p_1(\alpha), \ldots, p_m(\alpha)$ are distinct elements of $\mathbb{F}_p$. Since, $e_i(\alpha) = p_i(\alpha) \bmod p$, this means that $e_1(\alpha), \ldots, e_m(\alpha)$ are also distinct - we say that $\alpha \in \mathbb{F}_p$ is a 'good choice' of t. Now, let

$$g_\alpha(x) = f(x, x^{\alpha \bmod p}, x^{\alpha^2 \bmod p}, \ldots, x^{\alpha^{n-1} \bmod p}) = \sum_{i=1}^{m} c_i x^{e_i(\alpha)},$$

and p be the smallest prime greater than $\max\{d, \binom{m}{2} n\}$. Then, $e_i(\alpha) \leq d \cdot p \leq 2(mnd)^2$ implying that, degree of $g_\alpha(x)$ is bounded by $2(mnd)^2$. Hence, for any fixed choice of α we can interpolate $g_\alpha(x)$ by quering f at points $(x, x^{\alpha \bmod p}, x^{\alpha^2 \bmod p}, \ldots, x^{\alpha^{n-1} \bmod p})$ for $2(mnd)^2$ values of x from the underlying field $\mathbb{F}$. If $|F| < 2(mnd)^2$, work with a 'small' extension field $\mathbb{K} \supset \mathbb{F}$ that has $2(mnd)^2$ elements.

Once we interpolate $g_\alpha(x)$ for a good choice α, we know all the coefficients $c_1, \ldots, c_m$ of f, but we still do not know the monomials in f. The monomials are recovered in the following way: Since, $e_i(\alpha)$'s are distinct for $1 \leq i \leq m$, we can index the monomials of f by $e_i(\alpha)$'s. Suppose we want to know what is the degree of x_j occurring in the monomial of f that is indexed by $e_i(\alpha)$. Let $a \in \mathbb{F}$ be an element whose order is greater than d, i.e. $a^e \neq 1$ for $e < d$. (If $\mathbb{F}$ does not contain such an element, we work with a small extension field

$\mathbb{K}$. See exercise 1.). Define the polynomials,

$$g_{j,\alpha}(x) = f(x, x^{\alpha \bmod p}, \ldots, ax^{\alpha^{j-1} \bmod p}, \ldots, x^{\alpha^{n-1} \bmod p}) = \sum_{i=1}^{m} c_i a^{d_{ij}} x^{e_i(\alpha)},$$

We can interpolate $g_{j,\alpha}(x)$ in a way similar to $g_\alpha(x)$. The degree of x_j in the monomial of f that is indexed by $e_i(\alpha)$ is d_{ij}. We pick the coefficients of $x^{e_i(\alpha)}$ from $g_{j,\alpha}(x)$ and $g_\alpha(x)$ and divide them to obtain $a^{d_{ij}}$. Since we know a and $a^{d_{ij}}$, and the order of a is greater than d, we can find d_{ij} (by brute force search) using $O(d)$ operations over $\mathbb{F}$ (or, the extension field $\mathbb{K}$).

We need to address one final issue: How do we know that α is a good choice of t? We try out $k = \binom{m}{2} \cdot n$ distinct values of t from $\mathbb{F}_p$ (say, $\alpha_1, \ldots, \alpha_k$) and construct the polynomial $g_{\alpha_\ell}(x)$ for every choice $t = \alpha_\ell$ ($1 \leq \ell \leq k$). The one $g_\alpha(x)$ having maximum number of non-zero monomials clearly points out a 'good choice' of t.

The above discussion suggests the following algorithm. Here, p is a prime greater than $\max\{d, \binom{m}{2} \cdot n\}$, and $a \in \mathbb{F}$ has order greater than d.

Algorithm 1 Sparse polynomial interpolation

1. Pick $k = \binom{m}{2} \cdot n$ distinct elements, $\alpha_1, \ldots, \alpha_k$ from $\mathbb{F}_p$.
2. For each α_ℓ, interpolate the polynomial $g_{\alpha_\ell}(x) = f(x, x^{\alpha_\ell \bmod p}, x^{\alpha_\ell{}^2 \bmod p}, \ldots, x^{\alpha_\ell{}^{n-1} \bmod p})$.
3. Among $g_{\alpha_\ell}(x)$ $(1 \leq \ell \leq k)$, pick $g_\alpha(x) = \sum_{i=1}^{m} c_i x^{e_i(\alpha)}$ having maximum number of monomials.
4. For every $j \in [n]$, interpolate $g_{j,\alpha}(x) = f(x, x^{\alpha \bmod p}, \ldots, ax^{\alpha^{j-1} \bmod p}, \ldots, x^{\alpha^{n-1} \bmod p})$. Clearly, $g_{j,\alpha}(x) = \sum_{i=1}^{m} c_i a^{d_{ij}} x^{e_i(\alpha)}$.
6. For every $j \in [n], i \in [m]$, divide the coeff$(x^{e_i(\alpha)})$ in $g_{j,\alpha}$ and g_α to obtain $a^{d_{ij}}$.
7. Compute d_{ij} from $a^{d_{ij}}$ and a by brute force. Return $f = \sum_{i=1}^{m} c_i \prod_{j=1}^{n} x_j^{d_{ij}}$

Time complexity - It is clear from the above discussion that the algorithm runs in time polynomial in m, n and d. We leave the analysis of the exact time complexity as an exercise.

Exercises:

1. Show that if $|\mathbb{F}| > d^2$ then $\mathbb{F}$ contains an element whose order is greater than d.
2. Work out the precise time complexity of Algorithm 1.

References

[KS01] Adam Klivans and Daniel A. Spielman. Randomness efficient identity testing of multivariate polynomials. In *STOC*, pages 216–223, 2001.