An effective description of the roots of bivariates mod p^k and the related Igusa's local zeta function

Sayak Chakrabarti*
CS, Columbia University
New York, USA
sayaksc@gmail.com

Nitin Saxena
CSE, IIT Kanpur
Kanpur, India
nitin@cse.iitk.ac.in

ABSTRACT

Finding roots of a bivariate polynomial $f(x_1, x_2)$, over a prime field $\mathbb{F}_p$, is a fundamental question with a long history and several practical algorithms are now known. Effective algorithms for describing the roots modulo p^k, $k \geq 2$, for any general bivariate polynomial, were unknown until the present paper. The main obstruction is lifting the *singular* $\mathbb{F}_p$ roots to $\mathbb{Z}/p^k\mathbb{Z}$. Such roots may be numerous and behave unpredictably, i.e., they may or may not lift from $\mathbb{Z}/p^j\mathbb{Z}$ to $\mathbb{Z}/p^{j+1}\mathbb{Z}$.

We give the first algorithm to describe the roots of a bivariate polynomial over $\mathbb{Z}/p^k\mathbb{Z}$ in a practical way. Notably, when the degree of the polynomial is constant, our algorithm runs in deterministic time which is polynomial in $p + k$. This is a significant improvement over brute force, which would require exploring p^{2k} possible values. Our method also gives the first efficient algorithms for the following problems (which were also open): (1) efficiently representing all the (possibly infinitely-many) p-adic roots, and (2) computing the underlying Igusa's local zeta function. We also obtain a new, effective method to prove the rationality of this zeta function.

CCS CONCEPTS

• **Theory of computation** → *Algebraic complexity theory*; *Problems, reductions and completeness*; • **Computing methodologies** → **Algebraic algorithms**; Hybrid symbolic-numeric methods.

KEYWORDS

polynomial, prime-power, bivariate, p-adic, root-finding, root-counting, deterministic, Igusa, zeta function, tree

ACM Reference Format:
Sayak Chakrabarti and Nitin Saxena. 2023. An effective description of the roots of bivariates mod p^k and the related Igusa's local zeta function. In *International Symposium on Symbolic and Algebraic Computation 2023 (ISSAC 2023), July 24–27, 2023, Tromsø, Norway.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3597066.3597115

*Part of this work appears in the author's Master thesis at IIT Kanpur (e-link at [8]).

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
ISSAC 2023, July 24–27, 2023, Tromsø, Norway
© 2023 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM ISBN 979-8-4007-0039-2/23/07…$15.00
https://doi.org/10.1145/3597066.3597115

1 INTRODUCTION

Roots of polynomials in fields and rings have played an important role in mathematics and computer science for decades, with applications in a variety of topics. The roots of univariate polynomials modulo prime powers are relatively easy to find as we can find the roots in finite fields using factorization [1, 2, 7, 40, 51], after which we can efficiently lift the roots to modulo higher powers of p using the recent developments from [3, 22, 49]. However, even though we can factorize *multi*variate polynomials [35, 36], their roots usually do not correspond to a factor. Eg. even the irreducible polynomial $y - x$ has numerous roots! Such polynomials pose problems in root-finding as they have exponentially many roots in the base/prime field itself, and we can not quickly guess which root will lift to mod p^2. In our paper, we resolve this issue by giving an efficient algorithm to find roots modulo p^k for any prime p and k (given in unary); assuming that the degree d of the polynomial and the number n of the variables, are both small.

Prime field $\mathbb{F}_p$ and *p-adic integers* $\mathbb{Z}_p$ are unique factorization domains, and polynomials (above them) behave in an expected way. There are some algorithms to factorize polynomials in $\mathbb{Z}_p$ [6, 14, 27]. However, the properties in rings of characteristic as prime powers, which can be seen as a world between $\mathbb{F}_p$ and $\mathbb{Z}_p$, are still a mystery to us. There has been extensive work in this since the famous Hensel's lifting [30, 56, 57], where factors of polynomials are lifted from $\mathbb{F}_p$ to $\mathbb{Z}/p^k\mathbb{Z}$. Several variants of Hensel's lifting are available in various topics in algebra & number theory; but they fail when the polynomial does not factorize into coprime factors. This, when interpreted in terms of roots, means that it is difficult to lift *singular* roots. (Eg. $f = x_1^3 - px_2^2$, or $f = x_1^3 - p$, modulo p^2. Here, it is unclear how to even test the *existence* of roots; as the only $\mathbb{F}_p$-root here, $x_1 \equiv 0 \bmod p$, starts behaving unpredictably when 'lifted' modp^2.) There have been partially successful attempts to tackle this, and achieve factorization of *uni*variate polynomials mod p^k [11, 22, 52, 55]. Factorization mod p^k has only been solved until $k \leq 4$. Due to this, we can not use factorization, in any way, when finding roots mod p^k. Furthermore, due to the availability of 'exponentially' many factors, as well as roots, in these rings, its (data) structure has been of interest in mathematics and computer science. Eg. [17, 28, 46] analyze root-sets mod p^k, while [12, 21, 23, 41, 50] count the number of roots for a given polynomial. However, most of the works are restricted to univariate polynomials, as we did not have practical algorithms to find even *one* root of *bi*variate polynomials mod p^k. *This paper gives the first algorithm to find all the roots of a bivariate degree-d polynomial, over* $\mathbb{Z}/p^k\mathbb{Z}$ *resp.* $\mathbb{Z}_p$, *efficiently (for small d and varying p, k).* Our proofs extend to $n \geq 3$

in a natural way. However, due to the limited space we will only discuss $n = 2$.

Let us take a famous example as our algorithm's special-case—a *hyperelliptic curve*, given by the equation $y^2 + u(x)y + v(x) = 0$. Rational roots of (hyper)elliptic curves have been widely studied with several papers in this area [25, 38, 43, 45, 47, 53, 54]. Our algorithm can find all $(\mathbb{Z}/p^k\mathbb{Z})$-roots (resp. p-adic) of not only these, but general curves (that may have singularities). Even a *single* singular point forces us to explore its p^{k-1} possible lifts.

Root finding and counting have interesting applications in complexity theory too. We know that finding a solution to a system of constant-degree polynomials in any ring is NP-complete. Our algorithm solves a special case: where we want a common solution of *low*-variate, *low*-degree polynomials. Furthermore, root counting is a very hard problem. [24] showed that counting the number of roots of a multivariate polynomial of degrees as small as 3 is #P-complete, while [26] showed that modular root counting, over $\mathbb{F}_q$, is NP-hard for prime modulus other than the characteristic of the field. [59] also showed that the problem of computing Igusa's local zeta function is NP-complete even for $p = d = 2$ which can be shown from the arithmetization of SAT.

There has been extensive work on computing related zeta functions. [13, 29, 39, 42] compute the zeta function which encodes the size of a variety in finite fields, in time polynomial in the characteristic p. Improving this, say to $\mathrm{poly}(\log p)$, is a central open question. Our paper focuses on the regime of $\mathrm{poly}(p)$-time too, as the case of prime-power seems harder than finite fields (for instance, it has *no* known formulation of the famous Riemann Hypothesis).

In this paper, we are interested in another zeta function called the *Igusa's local zeta function* (Igusa's LZF), used to encode the number of roots of a polynomial modulo prime powers [32, 33]. [18] gave another proof for the rationality of this function. We give an alternate, more constructive proof by computing the Poincaré series; which can be seen as a generating function for point-counting. The Poincaré series for a polynomial f and a prime p is defined as $P_{f,p}(t) := \sum_{i=0}^{\infty} N_i(f) \cdot (p^{-n}t)^i$, where $t \in \mathbb{C}$, $|t| < 1$ and $N_i(f)$ refers to the number of roots of f mod p^i. Choosing $t = p^{-s}$, we denote $P_{f,p}(p^{-s}) = \sum_{i=0}^{\infty} N_i(f) \cdot (p^{-n}p^{-s})^i$. [34] showed a connection between Poincaré series and Igusa's LZF.

Despite the rationality being proved, explicit computation of this zeta function has remained a challenge. Root counting helps in computing this using the Poincaré series [19, 23, 59], but has been restricted to univariates. Giving the first algorithm, *we compute Igusa's local-zeta function for bivariates; thus, giving a new proof of its rationality as well.*

Due to space constraints, some of the proofs of the intermediate results, the methods for computing the Igusa's local-zeta function and the generalization to n-variates could not be included in this draft. However, the full version is available at [10]. The algorithmic techniques and proofs that have been omitted follow quite easily from the sequence of ideas presented here.

1.1 Previous work

There have been several works on finding roots of polynomials in rings. [49] gave an approach for finding roots of univariate polynomials modulo prime-powers in randomized polynomial time, which was greatly improved by [3, 48]. We are aware of only one work studying roots of *bi*variate polynomials mod p^k [50], where they count the total number of roots if the given polynomial $f(x, y)$ can be written as $f_1(x) + f_2(y)$, i.e., variable 'separated'. On the other hand, our algorithm does not require such assumptions.

Roots modulo p^k can be seen as an intermediate world between roots in $\mathbb{F}_p$ and roots in $\mathbb{Z}_p$. There have been papers to find roots of system of polynomials in certain finite fields [5, 31, 37, 44]. However, for finding $\mathbb{Z}_p$ roots, certain upper bounds given by N have been shown in [4, 15] which state that the existence of a solution mod p^N implies a $\mathbb{Z}_p$-root. Among the improved results, [15] showed $N \leq (nd)^{O(n^3 2^n)}$. [23] showed $N \leq d(\Delta + 1)$, where Δ is the discriminant-valuation; but it is only for univariate polynomials. We prove stronger structural results for the p-adic roots of bivariate polynomials, as discussed in Section 5.

Clearly, the literature suggests that roots behave "nicely" in $\mathbb{F}_p$ and $\mathbb{Z}_p$; but the properties in $\mathbb{Z}/p^k\mathbb{Z}$ are quite different for 'small' $k \geq 2$. [20, 58] have extensively explored the behavior of polynomials in these intermediate rings. In our paper, we essentially generalize the approach of [3] to non-trivially *reduce* root finding modulo p^k to the problem of solving a univariate multi-polynomial system.

1.2 Our results: Find roots in $\mathbb{Z}/p^k\mathbb{Z}$, $\mathbb{Z}_p$, and compute the Poincaré series

Our results give a new constructive understanding of the roots modulo p-power of bivariate systems. We use a new data-structure in the form of a *tree*, to view these roots with increasing exponent of the modulus. This tree data-structure, essentially, performs *desingularization* of roots— segregating them until they are non-singular —and lift to the required exponent of the prime.

Furthermore, we devise a new data-structure called *representative roots* to represent the exponentially many (resp. infinitely many p-adic) roots compactly. Our main ideas come from $n = 2$.

Theorem 1.1 (Bivariates). *Given $f \in (\mathbb{Z}/p^k\mathbb{Z})[x_1, x_2]$ of degree d, we can decide if a root of $f(x_1, x_2)$ exists, in deterministic* $\mathrm{poly}((k + d + p)^d)$ *time. If roots do exist, we can find and count all the roots (outputting them in a* compact *data structure).*

Based on Theorem 1.1, we give the following two corollaries, both of which were open problems. In a way, we bridge the gap between rings of the form $\mathbb{Z}/p^k\mathbb{Z}$ and $\mathbb{Z}_p$ by giving better bounds than existing works.

Corollary 1.2 (p-adic). *Given $f \in \mathbb{Z}[x_1, x_2]$ of degree d and the absolute value of its coefficients bounded above by $M > 0$, we can find all the p-adic-roots of f (in $\mathbb{Z}_p$) in deterministic* $\mathrm{poly}((\log M + d + p)^d)$ *time (i.e., output their k digits in a compact data structure).*

Corollary 1.3 (Local-zeta fn.). *Given $f \in \mathbb{Z}[x_1, x_2]$ of degree d and the absolute value of its coefficients bounded above by $M > 0$, we can compute the Poincaré series $P(t) =: A(t)/B(t)$ associated with f and a prime p, in deterministic* $\mathrm{poly}((\log M + d + p)^d)$ *time.*

The algorithm for computing p-adic roots, and the proof of Corollary 1.2 is described in Section 5. Based on this proof and simple power series computations similar to that of [23], we can compute the Igusa's zeta function for bivariates. This has been briefly described in Section 6.

There are major dissimilarities between roots of univariate and bivariate polynomials. However, n-variate polynomials, for 'small' $n \geq 2$, behave in a similar manner in our proof. We defer n-variates to the full version [10, Sec. 5.3], for the sake of simplicity. The algorithm for n-variates is quite similar to that of *bi*variate root-finding. Essentially, we reduce bivariate root finding to that of a system of univariates; whereas, for n-variates ($n \geq 3$), we reduce the problem to $(n-1)$-variate root-finding.

1.3 Difficulty of the problem

It is easy to lift a *non-singular* $\mathbb{F}_p$-root, i.e., root of $f \bmod p$ at which some first-order derivative of f is nonzero, to any $\mathbb{Z}/p^k\mathbb{Z}$ (see Theorem 2.1 and Corollary 2.3). In contrast, this paper is a significant advancement in the case when $\mathbb{F}_p$-roots are *singular*. It can be viewed as a reduction to non-singular roots, of a 'higher' dimension variety: which gets created while using the process of *lifting* (eg. extend a root (x_1, x_2) of $f \bmod p^j$ to mod p^{j+1} by perturbation). In our method, the dependence on p cannot be improved, as bivariates can have $\Omega(p)$ singular roots at each step of lifting.

Practicalities. Though our algorithm is slow for large degree *bi*variates or large primes, it is the first idea that works, for *small* degree d and prime p, much better than the brute-force algorithm. Our general algorithm is doubly-exponential in n (= number of variables), but, unsurprisingly the complexity is expected to be 'bad' in n as the problem of counting $\mathbb{F}_2$-roots (say, for a system with $d = 2$) is already NP-hard and even #P-hard [24, 26, 59].

1.4 Proof overview: Algorithms 1 & 2

Let us see the high-level idea in our main theorem, Theorem 1.1.

If (a_1, a_2) is a root of $f(x_1, x_2) \bmod p$, then we transform the polynomial to another polynomial given by $f(a_1 + px_1, a_2 + px_2)$. In order to find $\mathbb{F}_p$-roots of this polynomial in the next step, we remove the 'extra' p-powers by dividing by p^v; where $v := v_p(f(a_1 + px_1, a_2 + px_2))$ is the *val-multiplicity* and $v_p(\cdot)$ is the p-valuation. We define this step, of transforming the coordinates and subsequent division by the p-power, as the *lifting step* or *lifting of roots*. The polynomial will be modified at each step such that its $\mathbb{F}_p$-root returns a coordinate of the final (p-adic resp. $\mathbb{Z}/p^k\mathbb{Z}$) root. Notice that if (a_1, a_2) is an $\mathbb{F}_p$-root of $f(x_1, x_2)$, and after lifting, the polynomial becomes $\tilde{f}(x_1, x_2) := p^{-v} f(a_1 + px_1, a_2 + px_2)$ which has an $\mathbb{F}_p$-root (b_1, b_2), then $(a_1 + pb_1, a_2 + pb_2)$ is a root of $f(x_1, x_2) \bmod p^{v+1}$. The *uni*variate case of this lifting technique was developed in [3, 21].

However, it might happen that root (a_1, a_2) in the lifting process does *not* lift to higher powers of p; but some other root does lift, as illustrated by the following example.

Example 1.4. *Consider* $f(x_1, x_2) := x_1^3 - x_2^3 + 3x_2 - 3x_1 + 5$ *and* $p := 5$. $(1, 1)$ *and* $(2, 2)$ *are its* $\mathbb{F}_p$*-roots. Starting with the root* $(1, 1)$*, the process of lifting given by the transformation* $(x_1, x_2) \mapsto (1+5x_1, 1+5x_2)$ *and division by* 5*, yields the polynomial* $25x_1^3 - 25x_2^3 + 15x_1^2 - 15x_2^2 + 1$ *which does not have* $\mathbb{F}_5$*-roots. Although, restarting with the root as* $(2, 2)$ *yields the polynomial* $25x_1^3 - 25x_2^3 + 30x_1^2 - 30x_2^2 + 9x_1 - 9x_2 + 1$ *after lifting.* $(1, 0)$ *is now its* $\mathbb{F}_5$*-root!*

Thus, we iteratively loop over all the possible roots at each step, by fixing one variable, say x_1, with p-many possibilities, and finding the possible d-many (or p-many) values of x_2.

Val-multiplicity vs valuation. For a polynomial $f(\mathbf{x})$, we define the *effective polynomial* as $(f \bmod p)$, where the coefficients are in $\mathbb{F}_p$ (we can assume $f \bmod p$ is non-constant). Similarly, the *effective degree* d_1 of $f(\mathbf{x})$ is the degree of $(f \bmod p)$, while $d \geq d_1 \geq 1$ will be the total degree of f.

We define a *local root* of $f(\mathbf{x})$ as a root of the effective polynomial. For a local root $\mathbf{a} \in \mathbb{F}_p^2$, *local valuation* is defined as $v_p(f(\mathbf{a}))$. Recall: val-multiplicity of local root is defined as $v_p(f(\mathbf{a} + p\mathbf{x}))$, i.e., the minimum valuation of the coefficients of the polynomial thus formed; we sometimes shorten it to val-mult($\mathbf{a}$). Obviously, val-multiplicity is at most the local valuation.

Idea of Algorithm 1: Branching by val-multiplicities. As we have seen in Example 1.4, different $\mathbb{F}_p$ roots in steps of lifting can give rise to different val-multiplicities. Thus, we will view the algorithm as finding roots along a search *tree* (Fig.1). Note that there are at most dp local roots of f in $\mathbb{F}_p^2$.

The *nodes* of the tree contain the polynomials whose roots we are interested in finding from that point on. The *branches* arising from a node correspond to each local root $\mathbf{a} \in \mathbb{F}_p^2$. The *child*ren of this node will be the polynomials obtained from lifting by the local root in accordance to the branch, given by $f_{j+1}(\mathbf{x}) := p^{-v} f_j(\mathbf{a} + p\mathbf{x})$; where $v :=$ val-mult($\mathbf{a}$).

At depth j of the tree, the node contains the polynomial $f_j(\mathbf{x})$ that has been obtained by performing lifting j times on the original polynomial $f(\mathbf{x})$ using a contiguous sequence of local roots. Suppose we are at a node with $f_j(\mathbf{x})$ over $\mathbb{Z}/p^{k_j}\mathbb{Z}$, and consider the lifting according to root $\mathbf{a}$ with val-multiplicity $v_j := v_p(f_j(\mathbf{a} + p\mathbf{x}))$. The child of this node is $f_{j+1}(\mathbf{x}) := p^{-v_j} f_j(\mathbf{a} + p\mathbf{x})$, and we are interested in its roots over the ring $\mathbb{Z}/p^{(k_j - v_j)}\mathbb{Z}$. Thus, our target prime power reduces as we traverse down in the tree.

Theorem 2.1-(1) shows that the effective degree of the *newly* lifted polynomial will be at most the val-multiplicity of the local root considered, which in turn is $\leq d_1$ (= old effective degree). So, if val-multiplicity is $< d_1$, then the new effective degree reduces after lifting. Our algorithm's hard case is when a local root (a_1, a_2) has val-multiplicity d_1 (i.e., maximum possible). Then, the effective polynomial can be written as a "d_1-form", which is in the ideal $\langle x_1 - a_1, x_2 - a_2 \rangle^{d_1} \subseteq \mathbb{F}_p[\mathbf{x}]$ (Lemma 3.1). Next, we branch into a special val-multiplicity $= d_1$ part of the search tree, which requires a more complicated version of 'lifting steps' as we shall see.

To summarize, the degree reduction case of local root is where effective degree reduces ($v < d_1$ suffices, due to Theorem 2.1); while val-multiplicity d_1 case is where val-mult $v = d_1$.

Naively, the depth of this search tree can be $\Omega(k)$, as very few lifting steps may have degree reduction. Our second big idea is a way to overcome this basic obstruction (Algorithm 2).

At any given node, we consider the 'chains' that are responsible for the contiguous val-multiplicity d_1 branches. We store them in an array (D in Algo.1). This is done in the red part of the tree in Fig.1. After this is done, the search tree branches into the easier degree-reduction cases (branches with val-mult.$< d_1$); which is denoted by the green nodes (enclosed by the left rectangle in Fig.1).

So, in the tree, the lifting can either arise simply from the degree reduction cases, or can be a more complicated one where we transform the polynomial in Algorithm 2 after which we are guaranteed to branch into a degree reduction case.

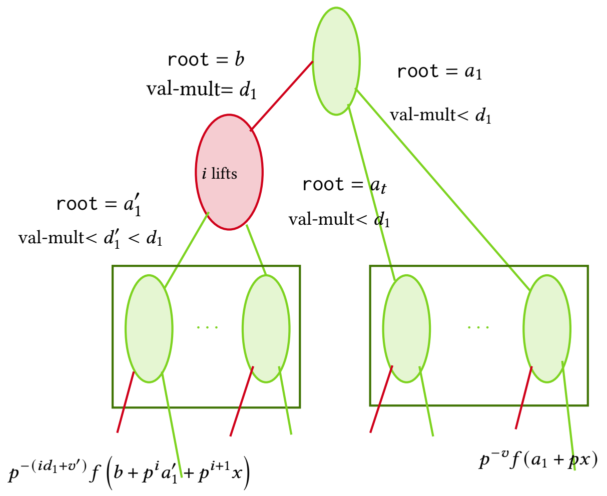

Figure 1: Branching along the search tree.

This has been schematically portrayed in Figure 1, where the red node is a more complicated transformation than the simple lifting of a local root (done in green node).

Tree depth and fanin. The crux of Algorithm 1 lies in our Theorem 2.1 which states that optimistically the effective degree reduces "most of the time". The algorithm ends when we either have exhausted the power of p, denoted by k, or when the effective degree becomes 1. The latter case (Theorem 2.1) gives a root and is essentially (p-adic) Hensel lifting on a *linear* polynomial [30].

Algorithm 1 ensures that the tree-depth is $O(d)$, while the more complicated Algorithm 2 ensures that the number of branches at every node is $O(k^2dp)$.

Hensel's Lifting. Given a non-singular root $\mathbf{a}$ of polynomial $h(\mathbf{x})$, we can lift it to modulo any p-power (like in Theorem 2.1-(2)), using a variant of p-adic Hensel's lifting [30]. Since $\mathbf{a}$ is a non-singular root, at least one of the first-order derivatives of $h(\mathbf{x})$ will not vanish. Corollary 2.3 implies that the val-multiplicity is then exactly 1, and in the next step of lifting, the effective polynomial will be *linear*. If this linear polynomial is of the form $m_1x_1 + m_2x_2 + m_0$, then (say) we can fix x_1 to any value in $[p]$ and find the corresponding unique value of x_2 to yield a root by simple lifting. For the next p-adic coordinate, after lifting, these m_1, m_2 (coefficients of x_1 resp. x_2) will not change; while m_0 might change. Thus, from Theorem 2.1-(2), we have the fact that the effective polynomial continues to stay linear, and we can fix the current-coordinate x_1 to find the corresponding x_2 every time; enabling us to lift to modulo any p-power (for arbitrary fixing of x_1 in this example).

Idea of Algorithm 2: Chain of d_1-forms. We create a process of 'removing' (or combining) contiguous val-multiplicity d_1 lifts, instead of brute-forcing over them in the search tree. This removal-process is guided by something called d_1*-forms* (Lemma 3.1), and will be subdivided into single and multiple val-multiplicity d_1 roots. (1) When *multiple* val-multiplicity roots exist, we show in Lemma 3.2 that the polynomial has a special form (namely d_1*-power*). We traverse these cases in a contiguous way. (2) Next we traverse over cases where val-multiplicity d_1 root is *unique*. At the end, we encounter lesser val-multiplicity roots and we can recursively call the root-finding algorithm. Overall, we find the lengths of these contiguous traversals, as well as the possibilities of the underlying d_1-powers resp. d_1-nonpowers. This is discussed in the latter-half of Section 3, by considering a dynamic basis-change on the variables $\mathbf{x}$, that guides the search for local roots well. In this process we *reduce* the root-finding of bivariate polynomials to that of a system of univariate polynomials, and employ the idea of [3] to find representative-roots of a univariate polynomial system. Finally, see Step 15 (Algorithm 1) for the consolidated transformation that we call *contiguous chain of val-multiplicity* d_1 *lifts*.

To summarize, in this case of successive val-mult=d_1 lifts, we show that the contiguous chains are few (i.e., polynomial in k, d, p), and every branch appearing from these chains ends in a degree reduction case. This is depicted in the search tree (Fig.1) as a red node; it 'jumps' over all the val-mult=d_1 cases (Sections 3–4) before branching to the green nodes. This bounds the tree-depth to $2d$.

Stopping condition, representative roots and root-counting. The algorithm terminates when either a root gets completely specified mod p^k, or when effective degree ≤ 1 (any of its roots can be Hensel lifted all the way to our required power of p), or when no root exists. In the third case, the root-set returned is just the emptyset ϕ, while in the first case it is a singleton.

For the second case, roots will be returned in terms of representative roots. Eg. when the lifted polynomial is zero modulo p^ℓ, any value in $\mathbb{Z}/p^\ell\mathbb{Z}$ is a root, and thus we return $*_1$ resp. $*_2$ for the coordinates x_1 resp. x_2, which represent the entire $\mathbb{Z}/p^\ell\mathbb{Z}$. The roots returned will be $(*_1, *_2)$, with the number of possibilities being $p^{2\ell}$. This will be termed as our usual *representative root.*

When the effective degree is 1— as we sketched before, we can fix one variable, say x_1, as a local root and find the value of x_2 (in each p-adic coordinate one by one). Even if only one variable is present in the linear form, say x_2, the other variable x_1 will still be free; so, for any given value of x_1, denoted by $*$, we can find the corresponding values of the local roots of x_2, thus, yielding a root of the polynomial modulo p^ℓ. Let us denote this function for determining x_2 from any value of x_1 by $c(\cdot)$, which simply finds each coordinate of x_2 using Hensel's lifting. Thus, the output can be denoted as $(*, c(*))$. The number of roots represented in this expression is p^ℓ. This type of representative root will be termed as *linear-representative.* (Note: In the presence of an offset, eg. $(r_1 + p^{\ell_1}*, r_2 + p^{\ell_2}\mu(*))$, this case can contribute more roots, so a more careful calculation is done in Section 6.)

2 EVOLUTION OF EFFECTIVE DEGREE DURING LIFTING STEPS

In this section, we analyze the effective degree at each step and look more closely as to when this decreases, or remains the same, by looking at the val-multiplicity of the local root during lifting. The proof idea is to analyze the monomials in terms of x_1 and x_2, and see how they behave after the transformation $(x_1, x_2) \mapsto (a_1 + px_1, a_2 + px_2)$ followed by division by appropriate power of p. This can be summed up by the following theorem.

Theorem 2.1 (Degree reduction). *For a polynomial $f(x_1, x_2) \in (\mathbb{Z}/p^k\mathbb{Z})[x_1, x_2]$, given an $\mathbb{F}_p^2$-root (a_1, a_2) of $f(x_1, x_2)$, let us denote $g(x_1, x_2) := p^{-v} f(a_1 + px_1, a_2 + px_2)$, where $v := v_p(f(a_1 + px_1, a_2 + px_2))$. Let the previous effective degree be $d_1 := \deg(f(x_1, x_2) \bmod p)$*

and current effective degree be $d_2 := \deg(g(x_1, x_2) \bmod p)$. *Then the following holds:*

(1) *If* $d_1 > 1$, *then* $d_2 \le v \le d_1$. *(So,* $d_2 = d_1$ *only if* $v = d_1$.*)*
(2) *If* $d_1 = 1$, *then* $d_2 = 1$.

Example 2.2. *Let us see how the effective degree could* reduce. *Consider* $f(x_1, x_2) = x_1^2 + x_2^3 \bmod p$. *This has degree* $d_1 = 3$. *Clearly,* $(0, 0)$ *is its root modulo* p. *So, apply the transformation* $(x_1, x_2) \mapsto (0 + px_1, 0 + px_2)$, *to get* $g(x_1, x_2) := p^{-2} f(px_1, px_2) = x_1^2 + px_2^3$, *which has effective degree* $d_2 = 2 = v < d_1$.

The proof of Theorem 2.1 relies on analyzing the partial derivatives in Taylor's expansion. Using this technique, we get a corollary on partial derivatives of $f(\mathbf{x})$, which motivates the inclusion of the term 'multiplicity' in our new concept of 'val-multiplicity'.

Corollary 2.3. *Local root* $\mathbf{a}$ *of* $f(\mathbf{x})$ *has val-multiplicity* $\ge v$, *if and only if* $p^{v-|\mathbf{i}|} \mid \frac{\partial_{\mathbf{x}^{\mathbf{i}}} f(\mathbf{a})}{\mathbf{i}!}$, *for all orders* $|\mathbf{i}| < v$.

Using Theorem 2.1, we get the idea of effective degree reduction. If root $(a_1, a_2) \in \mathbb{F}_p^2$ is such that $f(a_1 + px_1, a_2 + px_2) \not\equiv 0 \bmod p^{d_1}$, then we can repeat the appropriate transformation $(x_1, x_2) \mapsto (a_1 + px_1, a_2 + px_2)$, until the effective degree reduces to 1. Once this happens, we have a *compact description* of all its roots by Hensel lifting, as we can arbitrarily fix one variable and uniquely find the p-adic value of the other variable.

However, the problem arises when the root (a_1, a_2) is such that $f(a_1 + px_1, a_2 + px_2) \equiv 0 \bmod p^{d_1}$. In this case, the degree may not reduce, and we might need to lift k/d_1 many times. This is computationally infeasible, the search-tree becomes very deep/large, and takes time exponential in k/d_1. We tackle this case next.

3 STRUCTURE OF F VIA RANK OF LOCAL ROOTS OF VAL-MULT=D_1

We need to handle the challenge of our local root $\mathbf{a}$ of f having val-multiplicity $v = d_1$. Here, the effective degree may not reduce in the next step. We first show the structure of such $f(x_1, x_2)$.

Lemma 3.1 (d_1-form at $\mathbf{a}$). *If* $\mathbf{a} \in \mathbb{F}_p^2$ *is a root of* $f(\mathbf{x}) \bmod p$ *such that* $f(a_1 + px_1, a_2 + px_2) \equiv 0 \bmod p^{d_1}$, *where* d_1 *is the effective degree of* f, *then* $f(\mathbf{x})$ *is in the ideal* $\langle x_1 - a_1, x_2 - a_2\rangle^{d_1} \subset \mathbb{F}_p[\mathbf{x}]$.

In Lemma 3.1's situation, if $\mathbf{a}$ is unique, then using the structure of f we can easily find the root (eg. a simple search in $\mathbb{F}_p^2$), and lift without getting into multiple val-mult=d_1 branching. A serious obstruction arises when there are *several* local roots $\mathbf{a}$ of val-multiplicity = d_1. We will now show the extra special structure of such an $f(x_1, x_2)$.

Without loss of generality, let $\mathbf{0}$ be a local root of val-multiplicity = d_1. This means that $f \in \langle x_1, x_2\rangle^{d_1}$. If another local root $\mathbf{a} \ne \mathbf{0}$ exists with val-multiplicity = d_1, then we also have $f(\mathbf{x}) \in \langle x_1 - a_1, x_2 - a_2\rangle^{d_1}$. So, $f \in \langle x_1, x_2\rangle^{d_1} \cap \langle x_1 - a_1, x_2 - a_2\rangle^{d_1} \subset \mathbb{F}_p[\mathbf{x}]$. Then, we show f to be a *perfect-power*!

Lemma 3.2 (Two val-mult=d_1 roots). *For a polynomial* $f \in \mathbb{F}_p[x_1, x_2]$ *of degree* d_1, *iff* f *is in the ideal* $\langle x_1, x_2\rangle^{d_1} \cap \langle x_1 - a_1, x_2 - a_2\rangle^{d_1}$, *for some* $\mathbf{a} \ne \mathbf{0} \in \mathbb{F}_p^2$, *then we have* $f \equiv c \cdot (a_2x_1 - a_1x_2)^{d_1} \bmod p$, *where* $c \in \mathbb{F}_p^*$.

Essentially, this means f is d_1-th power of a linear polynomial iff rank of the val-mult=d_1 roots is two (i.e., multiple such roots). In the case of unique val-mult=d_1 root we will call the polynomial d_1*-nonpower-form*, while that for multiple val-mult=d_1 roots, we call the polynomial d_1*-power.*

Branching in d_1-nonpower-form. In this case, find the unique val-multiplicity d_1 root, and do the lifting step. There is no branching required.

Branching in d_1-power. Without loss of generality, the effective polynomial will be of the form $(a_2x_1 - a_1x_2)^{d_1}$. So, there are p roots (of val-mult=d_1), namely, (a_1t, a_2t) for any $t \in \mathbb{F}_p$. This leads to branching, which we will avoid by inventing a different strategy.

The first observation (Lemma 3.3) is that d_1-nonpower-form can not lead to a d_1-power. Thus, we deduce that whenever a contiguous chain of d_1-power lifting ends, then every d_1-form in the subsequent contiguous lifting steps is a d_1-nonpower-form.

Lemma 3.3 (Nonpower to power?). *If* f *is a* d_1*-nonpower-form having a single val-mult=*d_1 *root* $\mathbf{a}$, *then its lift* $p^{-d_1} f(\mathbf{a} + p\mathbf{x})$ *is not a* d_1*-power.*

So we mainly need to study the case: A d_1-power, say L^{d_1}, is followed by another d_1-power, say L'^{d_1}, in the next lifting step. Next, we unearth the structure that goes in the formation of L' after lifting the polynomial $L^{d_1} + \langle p\rangle$. This gives us the optimized bound on the branching of the red-nodes of the tree.

3.1 Structure of consecutive d_1-powers.

For a d_1-form, the effective polynomial $f(x_1, x_2) \bmod p$ will be of the form L^{d_1}, for some linear polynomial L (eg. $x_1 + x_2 + 1$). Without loss of generality, assume $\{L, x_2, 1\}$ to be of rank=3 (over $\mathbb{F}_p$). Let us rewrite f in the basis $\{L, x_2\}$, instead of $\{x_1, x_2\}$, denoted by $\tilde{f}(L, x_2)$ $(= f(\mathbf{x}))$. Since it is an invertible linear transformation, it now suffices to find roots of

$$\tilde{f} =: L^{d_1} + p \cdot L^{d_1-1} \cdot u_1(x_2) + p \cdot L^{d_1-2} \cdot u_2(x_2) + \cdots + p \cdot u_{d_1}(x_2)\,. \quad (1)$$

Lift d_1-power to d_1-power. Suppose that after lifting given by $p^{-d_1}\tilde{f}(pL, x_2)$, the effective polynomial is *again* a d_1-power; then it has to be the case that

$$\begin{aligned} L^{d_1} + L^{d_1-1} \cdot u_1(x_2) + L^{d_1-2} \cdot u_2(x_2)/p + \cdots + u_{d_1}(x_2)/p^{d_1-1} \\ \equiv (L + u_1(x_2)/d_1)^{d_1} \bmod p\,, \end{aligned} \quad (2)$$

for some univariate polynomials u_j's, such that Equation 2 is a perfect power of the linear polynomial $L + u_1(x_2)/d_1$. Consequently, those local roots a_2 for which the above system is satisfied, transform the previous 'root' L to $p\,(L + u_1(a_2)/d_1)$ in this lifting step.

Expanding the RHS, we also obtain equations, for $j \in [d_1]$, as

$$u_j(x_2) \equiv p^{j-1}\binom{d_1}{j} \cdot (u_1(x_2)/d_1)^j \bmod p^j\,. \quad (3)$$

Note: In the case where $p|d_1$, the above modulus can be further increased to clear away p-multiples from the denominator. In this fashion, we create a system of modular equations (in x_2) for the first step of lifting. Moving on, we consider the next lift.

Two consecutive d_1-power liftings. The effective polynomial after the first step was $(L + c(x_2))^{d_1}$. Let us look at the polynomials obtained before division by p^{d_1}. It was $(pL + pc(x_2))^{d_1} + \langle p^{d_1+1}\rangle$. Composing this with another lift of the same kind, the polynomial has to be of the form $(p(pL + pc(x_2)) + p^2\tilde{c}(x_2))^{d_1} + \langle p^{2d_1+1}\rangle$. This implies that we can directly lift $L \mapsto p^2L$, divide by p^{2d_1}, and find the value of $c(x_2) + \tilde{c}(x_2)$. So, Equation 2 can be replaced by the lift $p^{-2d_1}\tilde{f}(p^2L, x_2)$ equalling a d_1-power:

$$L^{d_1} + L^{d_1-1} \cdot u_1(x_2)/p + L^{d_1-2} \cdot u_2(x_2)/p^3 + \cdots + u_{d_1}(x_2)/p^{2d_1-1} \equiv (L + u_1(x_2)/pd_1)^{d_1} \bmod p\,. \quad (4)$$

Furthermore, we can write down the univariate modular equations like Equation 3 to find the root for x_2 that works in the lift.

In this way any i-length contiguous chain of d_1-power liftings, can be directly written as a system of univariate modular equations like Equation 3. It comes from the constraint that the lift $p^{-id_1}\tilde{f}(p^iL, x_2)$ has to equal a d_1-power mod p. Next, this system can be solved by adapting [3] to get the representative roots for the x_2 variable. Of course, on substituting this in x_2, we will know the final d_1-power L'^{d_1} that the contiguous i lifts yield.

How many consecutive d_1-powers? The length of this chain can be at most k/d_1. So, we go over all $i \le \lfloor k/d_1 \rfloor$. Iterating over them in decreasing order, we find all the possible ways of getting d_1-powers (before moving to other cases). This ensures that we do not miss any $(\mathbb{Z}/p^k\mathbb{Z})$-root of f in the search-tree.

Example 3.4. *Consider the polynomial $f(x_1, x_2) = x_1^2 \bmod p^k$. The d_1-power contiguous chain will be of length $k/2$; and each time $L = x_1$. The corresponding root will be $(p^{k/2} \cdot *_1, *_2)$.*

Notation for x_2 representatives. A problem arises when the representative for x_2 is $*_2$, i.e., x_2 can take any p-adic value. Eg. if we lift $f = L^{d_1} + p^{d_1}x_2^{d_1+1}$ (with free $x_2 = *_2$) then we get $g := L^{d_1} + x_2^{d_1+1}$. The degree of the new polynomial has now *increased*, which we never want to happen in our tree branchings. In order to prevent this, we preprocess the representative root $x_2 = *_2$ by increasing the precision by one coordinate. In other words, we consider the representative as $a + p \cdot *_2$, for $a \in \{0, \ldots, p-1\}$.

The following lemma shows that the effective degree never grows in lifting steps, in our algorithm.

Lemma 3.5 (Degree invariant). *The effective degree in each transformation described for d_1-forms is always d_1.*

Summing up. The structure discovered above gives a natural pseudocode that we describe in Algorithm 2. The contiguous val-mult=d_1 chain will have some d_1-powers, say i_1 many, followed by i_3 many d_1-nonpower forms, from which we have $i_1 + i_3 \le k/d_1$. The d_1-nonpower forms can not lead to d_1-powers again, due to Lemma 3.3. Also, to get the i_1 many d_1-powers, we need to use the univariate root-finding of [3] and get representatives R_1 for x_2 (in general L_2, independent of L_1). Going over each $i_1, i_3 \le \lfloor k/d_1 \rfloor$, and each of the representatives R_1, we compute the intermediate representative-roots R (and could continue with our recursion on the local roots with subsequent degree-reduction). This algorithm makes sure that, in the tree, we 'jump' the cases of val-mult= d_1 (effective degree) quickly, and reach the degree-reduction branchings.

4 THE ALGORITHM: PROOF OF THEOREM 1.1

Using the above ideas, we prove Theorem 1.1 by giving the complete algorithm to find roots of a bivariate polynomial modulo p^k.

4.1 Main algorithm for root-finding

Root-Find() in Algorithm 1 takes as *input*: the polynomial $f_j(x_1, x_2)$ and the number p^{k_j} ($k =: k_0$ initially). Main algorithm starts by calling Root-Find$(f(x_1, x_2), p^k)$. If there are valid roots, it outputs the set of roots $R \subseteq (\mathbb{Z}/p^k\mathbb{Z})^2$, otherwise returns ϕ.

Remove-d_1-Form() in Algorithm 2 eliminates intermediate lifts where effective degree does not decrease. It speeds-up the search for roots to higher precision coordinates, by jumping over contiguous cases of roots of val-multiplicity d_1. Remove-d_1-Form() outputs an *array* of: A linear transformation which can be used to jump over the val-multiplicity d_1 roots, or a linear-representative root.

Algorithm 1 Root Finding of $f_j(x_1, x_2) \bmod p^{k_j}$

```
procedure Root-Find($f_j(x_1, x_2), p^{k_j}$)
    if $k_j \le 0$ OR $f_j(x_1, x_2) \equiv 0 \bmod p^{k_j}$ then return $(*_1, *_2)$
    Define $d_1 := \deg(f_j \bmod p)$, $R := \phi$.
    if $d_1 = 1$ then
        return linear-representative $(*, c(*))$ or $(c(*), *)$, where $c(\cdot)$ is given by Hensel's Lifting.
    for $a_1 \in \{0, p-1\}$ do
        for $a_2$ such that $f_j(a_1, a_2) \equiv 0 \bmod p$ and val-mult(a)< $d_1$ do
            $f_{j+1}(x_1, x_2) := p^{-v} f_j(a_1 + px_1, a_2 + px_2)$, where $v := v_p(f_j(a_1 + px_1, a_2 + px_2))$.
            $S :=$ Root-Find$(f_{j+1}, p^{k_j - v})$ //aka green node in Fig.1
            $R := R \cup (\mathbf{a} + pS)$
    if val-multiplicity= $d_1$ root exists then
        $D :=$ Remove-$d_1$-Form$(f_j, p^{k_j})$ //aka red node in Fig.1
        for $(r_1 + p^{i_1}L_1, r_2 + p^{i_2}L_2, i_3) \in D$ do
            Write $f_j$ in basis $\{L_1, L_2\}$ to get $\tilde{f}_j(L_1, L_2) := f_j(x_1, x_2)$.
            Lift it to $\tilde{f}_j(L_1, L_2) := p^{-i_3 d_1} \cdot \tilde{f}_j(r_1 + p^{i_1}L_1, r_2 + p^{i_2}L_2)$.
            if $k_j - i_3 d_1 \le 0$ then
                The roots will be $(r_1 + p^{i_1} \cdot *_1, r_2 + p^{i_2} \cdot *_2)$ in $(L_1, L_2)$ basis.
                Consider the tuple $(r_1 + p^{i_1} \cdot *_1, r_2 + p^{i_2} \cdot *_2)$ and perform the inverse linear transformation from $(L_1, L_2)$ to $(x_1, x_2)$ on this tuple as a whole. Store this representative root (with two independent $*$'s) in a set $S$.
                $R := R \cup S$
            else
                For $\tilde{f}_j \bmod p^{k_j - i_3 d_1}$, find the val-mult $< d_1$ local roots and then recursively find all the roots; as done in Steps 6-10. Call this set $\tilde{R}$.
                For each root $(\tilde{r}_1, \tilde{r}_2) \in \tilde{R}$ of $\tilde{f}_j \bmod p^{k_j - i_3 d_1}$: consider $(r_1 + p^{i_1}\tilde{r}_1, r_2 + p^{i_2}\tilde{r}_2)$ and perform inverse linear transformation from $(L_1, L_2)$ to $(x_1, x_2)$ on them. Store these final roots (mod $p^{k_j}$) in a set $S$.
                $R := R \cup S$
    return $R$
```

4.2 Remove-d_1-Form( ) subroutine: Handling contiguous d_1-forms (aka red nodes in Fig.1)

In Algorithm 1, we do not want the lifting to go on for several recursion steps; since, the time complexity is exponential in the number of steps (=tree-depth). The favorable case is when the effective-degree reduces, e.g., when the val-multiplicity of local root is $< d_1$ (from Theorem 2.1). As we will see, the Remove-d_1-Form() subroutine ensures inside the red nodes (whose starting local root has val-multiplicity $= d_1$) that the effective degree reduces when we go down to its child. In this section we sketch the pseudocode based on the ideas developed in Section 3.1.

Data structure returned. In order to lift the contiguous d_1-forms, we return an array of tuples of the form $(a_1 + p^{i_1}L_1, a_2 + p^{i_2}L_2, i_3)$. This gives us information on jumping over the val-mult= d_1 roots by first covering the d_1-powers followed by d_1-nonpowers. This is done in a basis (L_1, L_2) of variables possibly different from (x_1, x_2). As in Equation 4, we form equations in terms of L_2 and find the roots, such that after lifting according to these (representative) roots, the effective polynomial will be $L_1^{d_1}$. Note that in each lifting according to the fixed part of the representative root, the linear polynomial will change by only a constant. Therefore, $\{1, L_1, L_2\}$ will also span the same space as that of $\{1, x_1, x_2\}$. So, given a root in (L_1, L_2) basis, we can recover the root in (x_1, x_2) basis uniquely.

With information from this tuple, we can do the following sequence of liftings in 'one-shot': i_1-steps of d_1-powers at first, followed by i_3-steps of d_1-nonpower-forms.

Pseudocode. Summing up, Remove-d_1-Form( ) 'jumps' over the intermediate local roots of val-multiplicity d_1 so that Root-Find( ) can continue to degree reducing cases (in Steps 6-10 of Algo.1).

The *input* is the polynomial and the prime-power, while the *output* is a tuple of linear polynomials, denoting intermediate representative-roots (over $(\mathbb{Z}/p^k\mathbb{Z})^2$).

Considering the degree reduction and the val-multiplicity d_1 cases, we get the final number of leaves in the search tree as $(\text{fanin})^{\text{depth}} = O((k^2dp)^{2d})$, which gives us the following theorem.

Theorem 4.1 (Correctness of Algorithm 1). *Given a polynomial $f \in \mathbb{Z}[x_1, x_2]$ of degree d, a prime p and an integer k. Algorithm 1 using Algorithm 2 as a subroutine, correctly returns all the roots* $\mathbf{a} \in (\mathbb{Z}/p^k\mathbb{Z})^2$ *of f mod p^k, in deterministic* $\text{poly}((k+d+p)^d)$ *time.*

5 COMPUTING P-ADIC ROOTS: PROOF OF COROLLARY 1.2

In this subsection, we give a bound for k_0 in terms of the degree d and the maximum absolute value M of the coefficients, such that finding a root modulo p^{k_0} would imply finding all representative (p-adic) $\mathbb{Z}_p$-roots of f. Let $f(x_1, x_2) \;=:\; \prod_{i=0}^{r} g_i(x_1, x_2)^{e_i}$, where $g_i(x_1, x_2)$'s are coprime over $\mathbb{Z}_p$. Even if f has some square-full factors (some e_i's are ≥ 2), we can eliminate them efficiently, by computing its gcd with the first-order derivatives. This will result in the new polynomial being of the form $\prod_{i=0}^{r} g_i(x_1, x_2)$, which we will call the *radical polynomial* $\text{rad}(f)$. The polynomial f and its radical $\text{rad}(f)$ have the same set of roots in $\mathbb{Z}_p$. Next, we bound the absolute-value of the coefficients of the radical.

Algorithm 2 Finding intermediate val-mult=d_1 roots in one-shot

```
1:  procedure Remove-$d_1$-Form($f(x_1, x_2), p^k$)
2:      Define $d_1 := \deg(f \bmod p)$, $R := \phi$.
3:      for $i_1 \in \{\lceil k/d_1 \rceil, \ldots, 0\}$ do
4:          $R_1 := \phi$
5:          Find the linear polynomial $L$ such that $f \equiv L^{d_1} \bmod p$.
            If $L$ is $x_2$-multiple, then set $L_2 := x_1$, otherwise set
            $L_2 := x_2$.
6:          Compute the (basis-change) polynomial $\tilde{f}$ such that
            $\tilde{f}(L, L_2) = f(x_1, x_2)$.
7:          Write $\tilde{f}$ as in Equation 4 and form (univariate, modular)
            equations like Equation 3 in terms of polynomials in $L_2$
            such that $i_1$-many contiguous $d_1$-powers exist (Section
            3.1).
8:          Find the representative-roots, in $\mathbb{Z}/p^{i_1 d_1}\mathbb{Z}$, of the system
            of equations formed in terms of $L_2$ (as in the previous
            step) using [3] and store them into $R_1$.
9:          for each representative-root $r_2 + p^{i_2} * \in R_1$ do
10:             Find linear polynomial $L_1$ obtained in the end, by
                substituting the representative in $L_2$, using the
                method of Section 3.1. Note: $i_2 \geq 1$ and $L_1$ has to be
                of the form $L + c$ for some integer $c$.
11:             Write $\tilde{f}(L, L_2)$ in basis $L_1, L_2$ given by $g(L_1, L_2) :=$
                $\tilde{f}(L, L_2)$.
12:             Lift $g(L_1, L_2) := p^{-i_1 d_1} \cdot g(p^{i_1} L_1\,,\, r_2 + p^{i_2} L_2)$
13:             for $i_3 \in \{\lceil k/d_1 \rceil - i_1, \ldots, 0\}$ do
14:                 if $\exists \mathbf{r}' \in (\mathbb{Z}/p^{i_3}\mathbb{Z})^2$ s.t. $g$ is a $d_1$-nonpower-form
    consecutively $i_3$-times then
15:                     In each precision $\mathbf{r}'$ is unique; so it can be
                        searched easily in the space $\mathbb{F}_p^2$.
16:                     $R_0 := (\,p^{i_1} r_1' + p^{i_1+i_3} L_1\,, r_2 + p^{i_2} r_2' +$
                        $p^{i_2+i_3} L_2\,, i_1 + i_3\,)$
17:                     if $R_0 \notin R$ then
18:                         $R := R \cup \{R_0\}$
19:                 else
20:                     break
21:     return R
```

Lemma 5.1 (Bound for p-adic radical). *If a polynomial f of degree d has the absolute-value of its coefficients bounded by M, then its radical has its coefficients bounded by $M^{O(d)}$.*

Therefore, without loss of generality we consider $f(x_1, x_2)$ to be square-free having absolute value of coefficients $\leq M^{O(d)}$, and continue with our algorithm of finding roots in $\mathbb{Z}_p$.

Representative roots in $\mathbb{Z}/p^k\mathbb{Z}$ vs roots in $\mathbb{Z}_p$. The return value of the algorithm, in the base-case, is either the representative root $(*_1, *_2)$ when the exponent of p required gets achieved (Step 2 of Algorithm 1), or linear-representative root $(*, c(*))$ (Steps 4-5 of Algorithm 1).

For large enough k, i.e., $k > k_0$, we want to show that if a representative root $(*_1, *_2)$ is returned, then the fixed part of the root is already a $\mathbb{Z}_p$-root. In the other case, for linear-representative roots, we can simply use Hensel lifting to lift to $\mathbb{Z}_p$-roots (or, to as much precision as we wish).

Discriminant. Let $f' := \partial_{x_2} f(x_1, x_2)$ be the first-order derivative of f. The resultant [16, Chapter 3] of f and f' w.r.t. x_2 is denoted $R(x_1) := \mathrm{Res}_{x_2}(f(x_1, x_2), f'(x_1, x_2))$, which is also one of the *discriminants* of f. $R(x_1)$ is not identically zero in $\mathbb{Z}_p$, as this would imply: f and its derivative have a common factor; contradicting the radical condition.

The roots of $R(x_1)$ given by $\hat{x}_1$ satisfy the condition that the univariate polynomial $f(\hat{x}_1, x_2)$ is square-full. Furthermore, given $\hat{x}_1$, we can easily find the values of x_2 (d-many), as it becomes the univariate root-finding problem over $\mathbb{Z}_p$ (solved in [3, 21, 23]).

Bound to distinguish $\mathbb{Z}_p$ roots. The main idea is to find a bound for the exponent of p such that each root returned using root-finding is either a linear-representative root, or a unique lift of this root is a $\mathbb{Z}_p$ root. A similar bound was achieved for univariate polynomials by [23]. However, the complications of lifting multivariate roots did not arise there, as every p-adic root corresponded to a factor.

Lemma 5.2 (Discriminant of radical). *Let $f \in \mathbb{Z}[x_1, x_2]$ be of degree d whose coefficients have absolute value bounded above by M. Let its radical polynomial be $g := \mathrm{rad}(f)$. The $\mathbb{Z}_p$-roots of $R(x_1) := \mathrm{Res}_{x_2}(g, g')$ are in one-one correspondence to the representative roots of $R(x_1) \bmod p^k$, for any $k \geq k_1 := \Theta(d^6 \log M)$.*

$\mathbb{Z}_p$-roots. Consider $g = \mathrm{rad}(f)$ and $k_1 = \Theta(d^6 \log M)$. Define $g_2(x_2) := \mathrm{Res}_{x_1}(g(x_1, x_2), R(x_1))$. Intuitively, roots x_2 of g_2 come from the roots x_1 of R. So, again applying [23, Theorem 20] on this univariate polynomial g_2, it suffices to compute its roots mod p^{k_2}, to compute its distinct p-adic roots; where k_2 is asymptotically $\log_p(p^{k_1 \cdot 2d^2 \cdot d}) = O(d^9 \log M)$.

Using the value of k_2 as above, we find roots of $g(x_1, x_2)$ from Root-Find(g, p^{k_2}). Let $(\tilde{a}_1, \tilde{a}_2)$ be the fixed-part of a root thus obtained. If $R(\tilde{a}_1) \equiv 0 \bmod p^{k_2}$, then the above argument, that defined k_2, ensures that $(\tilde{a}_1, \tilde{a}_2)$ does lift to a $\mathbb{Z}_p$-root of R, g_2, g and f (in this case uniquely).

Non-root of discriminant. Thus, the case left is that $R(\tilde{a}_1) \neq 0 \bmod p^{k_2}$. Consider the univariate polynomial $g(\tilde{a}_1, x_2)$. We know that its $\mathbb{Z}_p$-roots are different mod p^{k_2} and at most d many; one of which is $\tilde{a}_2$. Consider $g_1(x_2) := p^{-v} \cdot g(\tilde{a}_1\,, \tilde{a}_2 + x_2)$, where $v \geq 0$ is the p-valuation of $g(\tilde{a}_1\,, \tilde{a}_2 + x_2)$ as a polynomial over $\mathbb{Z}_p$. Note that x_2 divides g_1, but x_2^2 does not divide g_1 (mod p). Thus, 0 is a simple-root of g_1 and we can potentially Hensel lift it to p-adics.

To implement this formally, we need to increase the precision so that the extra p-factors can be removed from g. Note that if we assume $p \nmid g(\tilde{a}_1, x_2)$ then $v \leq k_2 + (k_2 - 1)(d-1) < d \cdot k_2$. Fix $k_0 := d \cdot k_2 = \Theta(d^{10} \log M)$. Now consider $\tilde{g}(\mathbf{x}) := p^{-v} \cdot g(\tilde{a}_1 + p^{k_2} x_1\,, \tilde{a}_2 + p^{k_2} x_2) \bmod p^{k_0}$. By the argument above, $\tilde{g} \bmod\ p$ is linear in x_2 (it is easier to see by substituting $x_1 = 0$). Thus, an extension of this root has to end up in some leaf of Root-Find(g, p^{k_0}) algorithm as say $(\tilde{a}_1', \tilde{a}_2')$; which will Hensel lift to p-adic integral root(s) due to the linear x_2 term in the lift.

Since the set of p-adic roots for f and g is the same, we could as well run Root-Find(f, p^{k_0}). This proves the following lemma.

Lemma 5.3 (p^{k_0} is p-adic). *Let $f \in \mathbb{Z}[x_1, x_2]$ be of degree d and having absolute-value of coefficients bounded by M. Each root represented in the leaves of the tree of Root-Find(f, p^{k_0}), for $k_0 := \Theta(d^{10} \log M)$, lifts to a $\mathbb{Z}_p$-root of $f(x_1, x_2)$.*

We further need the condition that the structure of this tree does not change with k for $k \geq k_0$. In order to show that, we prove the following lemma. Denote $R_1(x_1) := \mathrm{Res}_{x_2}(g, \partial_{x_2}(g))$ and $R_2(x_2) := \mathrm{Res}_{x_1}(g, \partial_{x_1}(g))$

Lemma 5.4 (Fix p-adic tree). *If a leaf of the tree given by Lemma 5.3 returns a representative root with the fixed part (a_1, a_2), that is not linear-representative, then $R_1(a_1) = R_2(a_2) = 0 \bmod p^k$. Moreover, (a_1, a_2) lifts to a unique root of f over $\mathbb{Z}_p$; and their number does not change as k grows beyond k_0.*

Also, the tree (Fig.1) in our algorithm does not change, and remains isomorphic, for $k \geq k_0$; except the leaf with the root $\mathbf{0}$.

The following examples should help illustrate the p-adic machinery more clearly.

Example 5.5. *Consider the polynomial $f = (x_1 - 1)(x_2 - 2) \bmod p^k$. The first step of our algorithm has to be $x_1 = 1$ or $x_2 = 2$. Considering the root $\mathbf{a} := (1, 3)$, the polynomial after lifting becomes $x_1(1 + px_2)$, which is an (effective) linear form; thus, a linear-representative root will be returned, which has x_2 as the free variable while x_1 will stay fixed to 1. This gives the leaf $\mathbf{r} := (1 + p\mu(*), 3 + p*)$, and a computable $\mathbb{Z}_p$-function $\mu(\cdot)$, which allows the p-adic lift of $\mathbf{a}$. In this case, $\mu = 0$.*

Example 5.6. *Now, consider $f(x_1, x_2) = (x_1 - px_2)(x_1 - 2px_2) \bmod p^k$. Lifting the root $\mathbf{a} := (0, 1)$ gives us $(x_1 - 1 - px_2)(x_1 - 2 - 2px_2)$, which is not yet effective linear. Choosing the next lifting-step around the root $(1, 0)$, the polynomial after lifting becomes $(x_1 - px_2)(-1 + px_1 - 2p^2 x_2)$, which is an (effective) linear polynomial; thus, a linear-representative root will be returned, corresponding to $(x_1 - px_2)$, which has x_2 as the free variable while x_1 depends on it. This gives the leaf $\mathbf{r} := (p + p^3\mu(*),\ 1 + p^2 *)$, and a computable $\mathbb{Z}_p$-function $\mu(\cdot)$, which allows the p-adic lift of $\mathbf{a}$. In this case, $\mu(w) := w$.*

Blowing up the root $\mathbf{0}$. There may be $\mathbb{Z}_p$-roots which can not be 'noticed' modulo p^{k_0}, because they are indistinguishable from $\mathbf{0}$. This is seen in the following example.

Example 5.7. *Consider the polynomial $x_1^3 + x_2^3 \bmod p^k$, for $p > 3$ and $3|k$. Some of its linear-representative roots are $(p^j + p^{j+1}*, -p^j + p^{j+1}\mu(*))$, for any $j < k/3$ and $\mu(w) := -w$. Also, $(p^{k/3}*_1, p^{k/3}*_2)$ is a non linear-representative root. It can lift to the p-adic root $\mathbf{0}$, but it can also lift to $(p^k, -p^k)$; which our algorithm could not explore due to the precision being only p^k.*

The following theorem completes the connection, of Algorithm 1, with *all* p-adic roots of f. Fundamentally, it scales up the roots by p^v-multiple, whenever possible, and creates a new data-structure for representatives in the leaves of the fixed tree modulo mod p^{k_0}, in Fig.1. It can also be seen as a way of further *blowing-up* the leaf of the fixed tree that gives the $\mathbf{0}$ root.

Theorem 5.8 (High val p-adic roots). *We can efficiently 'expand' the leaves, of the search tree, as follows:*

(1) Define a set of representative-roots $\mathcal{H}_v$, for $v \geq k_0$, s.t. for each

root $\mathbf{a} \in \mathcal{H}_v$ *,* $p^v\mathbf{a}$ *lifts to a p-adic root of f.*

(2) We can compute the fixed tree for $\mathcal{H}_{k_0}$ *efficiently by Algorithm 1. The other sets* $\mathcal{H}_v$*, for* $v > k_0$*, lift from the same representatives as in the leaves of* $\mathcal{H}_{k_0}$*; so we do not recompute them.*

Let (r_1', r_2') *be a p-adic root of f. Then,* $\exists v \geq 0$ *,* $\exists$ root $\mathbf{a} \in \mathcal{H}_v$ *lifting to* $\mathbf{a}'$ *, for which* $(r_1', r_2') = p^v\mathbf{a}'$*. In this sense, our fixed finite tree covers all (∞-many) p-adic roots of f.*

Lemmas 5.3-5.4 and Theorem 5.8 describe the p-adic nature of the tree and the representative roots, after the threshold bound of k_0. This finishes the proof of Corollary 1.2.

6 APPLICATION: IGUSA'S LOCAL ZETA FUNCTION (COROLLARY 1.3)

The essence of computing the Igusa's local zeta function is in finding all the roots modulo p^{k_0} for a large enough threshold k_0, such that representative roots in $\mathbb{Z}/p^{k_0}\mathbb{Z}$ are in one-one correspondence to those roots in $\mathbb{Z}/p^k\mathbb{Z}$ for any $k > k_0$. This turns out to be relatively easy in univariates (see [23]), however, for bivariates, a new kind of root, the blowing up of $\mathbf{0}$, comes into play.

Choosing $k_0 = \Omega(d^{10} \log M)$, as shown in Lemma 5.3, we compute the number of roots modulo p^k for $k \leq k_0$, using Algorithm 1. This can be performed in poly$(\log M, p)$ time. For $k > k_0$, we compute the roots in the following ways. The roots can be either a non linear-representative root, a linear-representative root, or a blow-up root. For non linear-representative roots, if the roots modulo p^{k_0} is of the form $(r_1 + p^{\ell_1}*_1, r_2 + p^{\ell_2}*_2)$, then the number of roots modulo p^k due to this will be $p^{k-\ell_1}.p^{k-\ell_2}$. For linear representative roots of the form $(r_1 + p^{\ell_1}*, r_2 + p^{\ell_2}\mu(*))$, it can be shown exploiting the structure of the polynomial, that the number of roots modulo p^k is $p^{\ell(k)}$, where $\ell(k)$ is a linear polynomial in k. Both these yield *geometric progressions* that can be summed up for values of k ranging from one to infinity. Finally, for the roots that blow up, we can precompute $\mathcal{H}_0$ from Theorem 5.8, which is independent of k, and multiply them by $(k - k_0)$ to give an arithmetic-geometric progression.

Thus, we compute $N_k(f)$ as a closed-form expression in k, such that $\sum_{k=k_0}^{\infty} N_k(f)(t/p^2)^k$ converges in $\mathbb{Q}(t)$. This completes the efficient computation of Igusa's local zeta function as a rational function in $t = p^{-s}$.

7 CONCLUSION AND FUTURE WORK

The aforementioned techniques of root-finding of bivariates modulo p^k naturally extend to n-variates with heavier complexity. The idea remains the same of lifting $\mathbf{x}$ to $\mathbf{a} + p\mathbf{x}$, and clearing out the extra powers of p, where $\mathbf{a}$ is a root modulo p. The algorithm works inductively, where root-finding of n-variates reduces to solving a system of polynomial equations over $(n-1)$ variables. Using some algebraic manipulations, and performing change of basis of variables while lifting, this yields an algorithm for finding a root of n-variate polynomials modulo p^k in time $O((k+d+p)^{(2d(n-1))^{n-1}})$. A more elaborate explanation of the ideas involved can be found in the full version of this paper on our homepage [8, 10].

Similarly, the $\mathbf{Z}_p$ roots of n-variates can be efficiently presented, which in turn leads to computing the Igusa's local zeta function. The power series expression obtained will again converge, thereby proving its rationality.

We leave the question of root-finding, for constant d and n, in polylog(p)-time open. Some progress has been made in [9] using deeper algebraic-geometric methods.

ACKNOWLEDGMENTS

We thank Ashish Dwivedi for useful discussions on many related problems in this area. We thank the anonymous reviewers whose feedback improved our presentation. N.S. thanks the funding support from DST-SERB (CRG/2020/000045) and N.Rama Rao Chair.

REFERENCES

[1] Elwyn R Berlekamp. 1967. Factoring polynomials over finite fields. *Bell System Technical Journal* 46, 8 (1967), 1853–1859.

[2] Elwyn R Berlekamp. 1970. Factoring polynomials over large finite fields. *Mathematics of computation* 24, 111 (1970), 713–735.

[3] Jérémy Berthomieu, Grégoire Lecerf, and Guillaume Quintin. 2013. Polynomial root finding over local rings and application to error correcting codes. *Applicable Algebra in Engineering, Communication and Computing* 24, 6 (2013), 413–443.

[4] BJ Birch and K McCann. 1967. A Criterion for the p-adic Solubility of Diophantine Equations. *The Quarterly Journal of Mathematics* 18, 1 (1967), 59–63.

[5] Andreas Björklund, Petteri Kaski, and Ryan Williams. 2019. Solving systems of polynomial equations over GF (2) by a parity-counting self-reduction. In *46th International Colloquium on Automata, Languages, and Programming (ICALP), 2019, Patras, Greece (LIPIcs, Vol. 132).* Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 26:1–26:13.

[6] David G. Cantor and Daniel M. Gordon. 2000. Factoring Polynominals over p-Adic Fields. In *Algorithmic Number Theory, 4th International Symposium, ANTS-IV, Leiden, Netherlands (Lecture Notes in Computer Science, Vol. 1838).* Springer, 185–208.

[7] David G Cantor and Hans Zassenhaus. 1981. A new algorithm for factoring polynomials over finite fields. *Math. Comp.* 36, 154 (1981), 587–592.

[8] Sayak Chakrabarti. 2022. *Multivariate polynomials modulo prime powers: their roots, zeta-function and applications.* Master's thesis. CSE, IIT Kanpur, India.

[9] Sayak Chakrabarti, Ashish Dwivedi, and Nitin Saxena. 2022. *Solving polynomial systems over non-fields and applications to modular polynomial factoring.* Technical Report. CSE, IIT Kanpur.

[10] Sayak Chakrabarti and Nitin Saxena. 2022. *An effective description of the roots of multivariates mod* p^k *and the related Igusa's local zeta function.* Technical Report. CSE, IIT Kanpur.

[11] Howard Cheng and George Labahn. 2001. Computing all factorizations in $Z_n[x]$. In *Proceedings of the 2001 International Symposium on Symbolic and Algebraic Computation (ISSAC), Ontario, Canada.* ACM, 64–71.

[12] Qi Cheng, Shuhong Gao, J Maurice Rojas, and Daqing Wan. 2019. Counting roots for polynomials modulo prime powers. *The Open Book Series* 2, 1 (2019), 191–205.

[13] Qi Cheng, J Maurice Rojas, and Daqing Wan. 2020. Computing zeta functions of large polynomial systems over finite fields. *arXiv preprint arXiv:2007.13214* (2020), 1–10.

[14] Alexander Leonidovich Chistov. 1987. Efficient factorization of polynomials over local fields. *Doklady Akademii Nauk* 293, 5 (1987), 1073–1077.

[15] Alexander L Chistov. 2021. An Effective Algorithm for Deciding Solvability of a System of Polynomial Equations over p-adic Integers. *Algebra i Analiz* 33, 6 (2021), 162–196.

[16] David Cox, John Little, and Donal O'Shea. 2013. *Ideals, varieties, and algorithms: an introduction to computational algebraic geometry and commutative algebra.* Springer Science & Business Media.

[17] Bruce Dearden and Jerry Metzger. 1997. Roots of polynomials modulo prime powers. *European Journal of Combinatorics* 18, 6 (1997), 601–606.

[18] Jan Denef. 1984. The rationality of the Poincaré series associated to the p-adic points on a variety. *Invent. math* 77, 1 (1984), 1–23.

[19] Jan Denef and Kathleen Hoornaert. 2001. Newton polyhedra and Igusa's local zeta function. *Journal of number Theory* 89, 1 (2001), 31–64.

[20] Ashish Dwivedi. 2023. *Polynomials over composites: Compact root representation via ideals and algorithmic consequences.* Ph. D. Dissertation. CSE, IIT Kanpur, India.

[21] Ashish Dwivedi, Rajat Mittal, and Nitin Saxena. 2019. Counting Basic-Irreducible Factors Mod p^k in Deterministic Poly-Time and p-Adic Applications. In *Proceedings of 34th Computational Complexity Conference (CCC 2019).* Springer, 15:1–15:29.

[22] Ashish Dwivedi, Rajat Mittal, and Nitin Saxena. 2021. Efficiently factoring polynomials modulo p^4. *Journal of Symbolic Computation* 104 (2021), 805–823.

[23] Ashish Dwivedi and Nitin Saxena. 2020. Computing Igusa's local zeta function of univariates in deterministic polynomial-time. *Open Book Series* 4, 1 (2020), 197–214.

[24] Andrzej Ehrenfeucht and Marek Karpinski. 1990. *The computational complexity of (xor, and)-counting problems.* International Computer Science Inst.

[25] Pierrick Gaudry and Robert Harley. 2000. Counting points on hyperelliptic curves over finite fields. In *International Algorithmic Number Theory Symposium.* Springer, 313–332.

[26] Parikshit Gopalan, Venkatesan Guruswami, and Richard J Lipton. 2008. Algorithms for modular counting of roots of multivariate polynomials. *Algorithmica* 50, 4 (2008), 479–496.

[27] Jordi Guàrdia, Enric Nart, and Sebastian Pauli. 2012. Single-factor lifting and factorization of polynomials over local fields. *Journal of Symbolic Computation* 47, 11 (2012), 1318–1346.

[28] Aditya Gulati, Sayak Chakrabarti, and Rajat Mittal. 2021. On Algorithms to Find p-ordering. In *Conference on Algorithms and Discrete Applied Mathematics.* Springer, 333–345.

[29] David Harvey. 2015. Computing zeta functions of arithmetic schemes. *Proceedings of the London Mathematical Society* 111, 6 (2015), 1379–1401.

[30] Kurt Hensel. 1918. Eine neue Theorie der algebraischen Zahlen. *Mathematische Zeitschrift* 2, 3 (1918), 433–452.

[31] M-D Huang and Y-C Wong. 1999. Solvability of systems of polynomial congruences modulo a large prime. *computational complexity* 8, 3 (1999), 227–257.

[32] Jun-ichi Igusa. 1974. Complex powers and asymptotic expansions. I. Functions of certain types. *Journal für die reine und angewandte Mathematik* 0268_0269 (1974), 110–130. http://eudml.org/doc/151455

[33] Jun-Ichi Igusa. 1977. Some observations on higher degree characters. *American Journal of Mathematics* 99, 2 (1977), 393–417.

[34] Jun-ichi Igusa. 2007. *An introduction to the theory of local zeta functions.* Vol. 14. American Mathematical Soc.

[35] Erich Kaltofen. 1982. A polynomial-time reduction from bivariate to univariate integral polynomial factorization. In *23rd Annual Symposium on Foundations of Computer Science (FOCS 1982).* IEEE, 57–64.

[36] Erich Kaltofen. 1985. Polynomial-time reductions from multivariate to bi-and univariate integral polynomial factorization. *SIAM J. Comput.* 14, 2 (1985), 469–489.

[37] Neeraj Kayal. 2005. Solvability of a system of bivariate polynomial equations over a finite field. In *International Colloquium on Automata, Languages, and Programming.* Springer, 551–562.

[38] Kiran S Kedlaya. 2001. Counting points on hyperelliptic curves using Monsky-Washnitzer cohomology. *Journal of the Ramanujan Mathematical Society* 16, 4 (2001), 323–338.

[39] Kiran S Kedlaya. 2004. Computing zeta functions via p-adic cohomology. In *International Algorithmic Number Theory Symposium.* Springer, 1–17.

[40] Kiran S Kedlaya and Christopher Umans. 2011. Fast polynomial factorization and modular composition. *SIAM J. Comput.* 40, 6 (2011), 1767–1802.

[41] Leann Kopp, Natalie Randall, Joseph Rojas, and Yuyu Zhu. 2020. Randomized polynomial-time root counting in prime power rings. *Math. Comp.* 89, 321 (2020), 373–385.

[42] Alan GB Lauder. 2006. A recursive method for computing zeta functions of varieties. *LMS Journal of Computation and Mathematics* 9 (2006), 222–269.

[43] Frank Lehmann, Markus Maurer, Volker Müller, and Victor Shoup. 1994. Counting the number of points on elliptic curves over finite fields of characteristic greater than three. In *International Algorithmic Number Theory Symposium.* Springer, 60–70.

[44] Daniel Lokshtanov, Ramamohan Paturi, Suguru Tamaki, Ryan Williams, and Huacheng Yu. 2017. Beating brute force for systems of polynomial equations over finite fields. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms.* SIAM, 2190–2202.

[45] Kazuto Matsuo, Jinhui Chao, and Shigeo Tsujii. 2002. An improved baby step giant step algorithm for point counting of hyperelliptic curves over finite fields. In *International Algorithmic Number Theory Symposium.* Springer, 461–474.

[46] Davesh Maulik. 2001. Root sets of polynomials modulo prime powers. *Journal of Combinatorial Theory, Series A* 93, 1 (2001), 125–140.

[47] Alfred J Menezes, Scott A Vanstone, and Robert J Zuccherato. 1993. Counting points on elliptic curves over $\mathbb{F}_{2^m}$. *Mathematics of computation* 60, 201 (1993), 407–420.

[48] Vincent Neiger, Johan Rosenkilde, and Éric Schost. 2017. Fast computation of the roots of polynomials over the ring of power series. In *Proceedings of the 2017 ACM on International Symposium on Symbolic and Algebraic Computation.* 349–356.

[49] Peter N Panayi. 1995. *Computation of Leopoldt's P-adic regulator.* Ph. D. Dissertation. University of East Anglia, Norwich, England.

[50] Caleb Robelle, J Maurice Rojas, and Yuyu Zhu. 2021. Sub-Linear Point Counting for Variable Separated Curves over Prime Power Rings. *arXiv preprint arXiv:2102.01626* (2021), 18.

[51] Lajos Rónyai. 1987. Factoring polynomials over finite fields. In *28th Annual Symposium on Foundations of Computer Science (FOCS 1987).* IEEE, 132–137.

[52] Ana Sălăgean. 2005. Factoring polynomials over Z4 and over certain Galois rings. *Finite fields and their applications* 11, 1 (2005), 56–70.

[53] Takakazu Satoh. 2002. On p-adic point counting algorithms for elliptic curves over finite fields. In *International Algorithmic Number Theory Symposium.* Springer, 43–66.

[54] René Schoof. 1995. Counting points on elliptic curves over finite fields. *Journal de théorie des nombres de Bordeaux* 7, 1 (1995), 219–254.

[55] Carlo Sircana. 2017. Factorization of polynomials over $\mathbb{Z}/(p^n)$. In *Proceedings of the 2017 ACM on International Symposium on Symbolic and Algebraic Computation.* 405–412.

[56] Hans Zassenhaus. 1969. On hensel factorization, I. *Journal of Number Theory* 1, 3 (1969), 291–311.

[57] Hans Zassenhaus. 1978. A remark on the Hensel factorization method. *Math. Comp.* 32, 141 (1978), 287–292.

[58] Yuyu Zhu. 2020. *Trees, point counting beyond fields, and root separation.* Ph. D. Dissertation. Texas A&M University, USA.

[59] WA Zuniga-Galindo. 2003. Computing Igusa's local zeta functions of univariate polynomials, and linear feedback shift registers. *Journal of Integer Sequences* 6 (2003), 36.