From Approximate Factorization to Root Isolation with Application to Cylindrical Algebraic Decomposition

Kurt Mehlhorn

Max Planck Institute for Informatics, Saarbrücken, Germany

Michael Sagraloff

Max Planck Institute for Informatics, Saarbrücken, Germany

Pengming Wang

Max Planck Institute for Informatics, Saarbrücken, Germany

Abstract

We present an algorithm for isolating all roots of an arbitrary complex polynomial p that also works in the presence of multiple roots provided that (1) the number of distinct roots is given as part of the input and (2) the algorithm can ask for arbitrarily good approximations of the coefficients of p. The algorithm outputs pairwise disjoint disks each containing one of the distinct roots of p and the multiplicity of the root contained in the disk. The algorithm uses approximate factorization as a subroutine. For the case where Pan's algorithm [36] is used for the factorization, we derive complexity bounds for the problems of isolating and refining all roots, which are stated in terms of the geometric locations of the roots only. Specializing the latter bounds to a polynomial of degree d and with integer coefficients of bitsize less than τ, we show that $\tilde{O}(d^3 + d^2\tau + d\kappa)$ bit operations are sufficient to compute isolating disks of size less than $2^{-\kappa}$ for all roots of p, where κ is an arbitrary positive integer.

In addition, we apply our root isolation algorithm to a recent algorithm for computing the topology of a real planar algebraic curve specified as the zero set of a bivariate integer polynomial and for isolating the real solutions of a bivariate polynomial system. For polynomials of degree n and bitsize τ, we improve the currently best running time from $\tilde{O}(n^9\tau + n^8\tau^2)$ (deterministic) to $\tilde{O}(n^6 + n^5\tau)$ (randomized) for topology computation and from $\tilde{O}(n^8 + n^7\tau)$ (deterministic) to $\tilde{O}(n^6 + n^5\tau)$ (randomized) for solving bivariate systems.

Key words: root isolation, root refinement, curve analysis, bivariate polynomial system, complexity analysis, cylindrical algebraic decomposition

1. Introduction

Root isolation is a fundamental problem of computational algebra and numerical analysis [23, 25, 32, 35, 53]. Given a univariate polynomial p with complex coefficients and possibly multiple roots, the goal is to compute disjoint disks in the complex plane such that each disk contains exactly one root and the union of all disks covers all roots. We assume the existence of an oracle that can be asked for rational approximations of the coefficients of arbitrary precision. In particular, coefficients may be transcendental. Note that non-rational coefficients can never be learned exactly in finite time.

In this generality, the problem is unsolvable. This is a consequence of the numerical halting problem [30, 54]. We give an example of a polynomial of degree three, for which, in the input model above, no finite algorithm can distinguish between the case of two or three distinct roots. When the coefficient oracle is asked for coefficients with precision L, it returns 1, α_L, β_L, and 2, where α_L and β_L are rational, the polynomial $p_L(x) = x^3 + \alpha_L x^2 + \beta_L x + 2$ has three distinct roots, and $\left|(-2\sqrt{2}-1) - \alpha_L\right| \leq 2^{-L}$ and $\left|\beta_L - (2+2\sqrt{2})\right| \leq 2^{-L}$. Observe that these answers of the oracle are consistent with the polynomials p_L and p, where $p(x) = (x-\sqrt{2})^2(x+1) = x^3 + (-2\sqrt{2}-1)x^2 + (2+2\sqrt{2})x + 2$. The former polynomial has three distinct roots and the latter polynomial has two distinct roots. Assume that the algorithm stops after asking for coefficients with precision L. If it outputs "two distinct roots", the oracle can claim that the input polynomial is p_L, if it outputs "three distinct roots", the oracle can claim that the input polynomial is p. In either case, the output is incorrect.

The example shows that the problem needs to be restricted. In addition to our assumption that the coefficients of our input polynomial p are provided by coefficient oracles, we further assume that *the number k of distinct roots* is also given. [1] For polynomials with algebraic coefficients, the number k can be computed via symbolic methods. In general, knowledge of k requires additional higher-level considerations. [2] We would like to explain why knowledge of k is nevertheless a reasonable assumption: Root isolation is a key ingredient in the computation of a CAD (cylindrical algebraic decomposition) for a set of multivariate polynomials and, in particular, for computing the topology of algebraic curves and surfaces. In these applications, one has to deal with polynomials with multiple roots and algebraic coefficients; the coefficients are easily approximated to an arbitrary precision. In addition, the number of distinct roots is readily available from an algebraic precomputation (e.g. computation of a subresultant sequence, triangular decomposition). We now give an overview of our algorithm, our results, and related work.

Root Isolation: We fix some definitions which are used throughout the presentation: Let $p(x) = \sum_{i=0}^{n} p_i x^i \in \mathbb{C}[x]$, with $1/4 \leq |p_n| \leq 1$, [3] be a complex polynomial with k distinct roots $z_1, \ldots, z_k$. For $i = 1, \ldots, k$, let $m_i := \operatorname{mult}(z_i, p)$ be the *multiplicity* of z_i, and let $\sigma_i := \sigma(z_i, p) := \min_{j \neq i} |z_i - z_j|$

Email addresses: mehlhorn@mpi-inf.mpg.de (Kurt Mehlhorn), msagralo@mpi-inf.mpg.de (Michael Sagraloff), s9pewang@stud.uni-saarland.de (Pengming Wang).

[1] An alternative restriction is to be content with the computation of well-separated clusters of roots, i.e., the computation of disks Δ_i and multiplicities m_i such that D_i contains exactly m_i roots counted with multiplicity, $\sum_i m_i$ is equal to the degree of the polynomial, and substantially enlarged disks are disjoint. Our algorithm also applies to this version of the problem. We come back to it in Section 2.4.

[2] One might think of a polynomial p with arbitrary real (even transcendental) coefficients for which we are aware of a theoretical argument showing that p has k distinct roots.

[3] The additional requirement for the leading coefficient p_n yields a simpler presentation. Notice that, for general values p_n, we first have to multiply the polynomial p by some 2^t, with $t \in \mathbb{Z}$, such that $2^t \cdot |p_n|$ is contained in $[1/4, 1]$.

be the *separation* of z_i from the other roots of p. Then, our algorithm outputs isolating disks $\Delta_i = \Delta(\tilde{z}_i, R_i)$ for the roots z_i and the corresponding multiplicities m_i. The radii satisfy $R_i < \frac{\sigma_i}{64n}$, and hence the center $\tilde{z}_i$ of Δ_i approximates z_i to an error of less than $\frac{\sigma_i}{64n}$. If the number of distinct roots of p differs from k, we make no claims about termination and output.

The coefficients of p are provided by oracles. That is, on input L, the oracle essentially returns binary fraction approximations $\tilde{p}_i$ of the coefficients p_i such that $\left\| p - \sum_{i=0}^{n} \tilde{p}_i x^i \right\| \le 2^{-L} \|p\|$. Here, $\|p\| := \|p\|_1 = |p_0| + \ldots + |p_n|$ denotes the *one-norm* of p. The details are given in Section 2.1. The assumption that the coefficients are given through oracles is standard in computational real analysis [30] and numerical analysis [23, 24, 32], and is used in previous papers on approximate factorization and root isolation [36, 46].

Many algorithms for approximate factorization and root isolation are known, see [17, 32] for a survey. The algorithms can be roughly split into two groups: there are iterative methods for simultaneously approximating all roots (or a single root if a sufficiently good approximation is already known), and there are subdivision methods that start with a region containing all the roots of interest, subdivide this region according to certain rules, and use inclusion- and exclusion-predicates to certify that a region contains exactly one root or no root. Prominent examples of the former group are the Aberth-Ehrlich method (used for MPSOLVE [5]) and the Weierstrass-Durand-Kerner method. These algorithms work well in practice and are widely used. However, a complexity analysis and global convergence proof is missing. Prominent examples of the second group for isolating all complex roots are the Bolzano method [8, 44] and the splitting circle method [36, 46]. There are also methods, e.g., the Descartes, Sturm, and continued fraction methods, for isolating the real roots of a real polynomial. Among the subdivision methods, the splitting circle method is asymptotically the best. It was introduced by Schönhage [46] and later considerably refined by Pan [36]. An implementation of the splitting circle method in the computer algebra system Pari/GP is available [22]. None of the algorithms mentioned deals specifically with multiple roots. For square-free polynomials, i.e, the case $k = n$, the subdivision methods guarantee root isolation. For integral polynomials, a square-free decomposition can be computed [51]. Alternatively, separation bounds [53, Section 6.7] can be used to guarantee isolation in the presence of multiple roots. Johnson [27], Cheng et. al. [11], and Strzebonski and Tsigaridas [50] discuss root isolation for polynomials with algebraic coefficients. However, they have to consider worst case separation bounds in the presence of multiple roots, an approach which is considered to be not practical.

Strzebonski [49] presents an algorithm that deals with multiple roots in our setting. However, it has heuristic steps. The algorithm in [33] can cope with at most one multiple root and needs to know the number of distinct complex roots as well as the number of distinct real roots. Algorithms for root refinement, e.g., Newton-Raphson iteration, compute arbitrary good approximations to roots once a good initial approximation is known. Generalization to clusters of roots are provided by [19, 52].

Our algorithm has a simple structure. It combines mainly known techniques. Our contribution is the right assembly into an algorithm, our novel clustering step, and the complexity analysis. We first use any algorithm (e.g. [5, 36, 46, 44]) for approximately factorizing the input polynomial. It is required that it can be run with different levels of precision, and that, for any given integer b, it returns approximations $\hat{z}_1$ to $\hat{z}_n$ for the roots of p such that

$$\left\| p - p_n \prod_{1 \le j \le n} (x - \hat{z}_j) \right\| \le 2^{-b} \|p\| . \tag{1}$$

In a second step, we partition the root approximations $\hat{z}_1$ to $\hat{z}_n$ into clusters $C_1, C_2 \ldots$ based on geometric vicinity. If the number of clusters is less than k, we increase the precision, and refactor.

The difficulty of the clustering step lies in the fact that the amounts by which roots will move after a perturbation of the coefficients (recall that, in our input model, we only see perturbations of the true coefficients) depends heavily on the multiplicity of the root. We enclose each cluster C_i in a disk $D_i = \Delta(\tilde{z}_i, r_i)$ and make sure that the disks are pairwise disjoint and that the radii r_i are not "too small" compared to the pairwise distances of the centers $\tilde{z}_i$.[4] In a third step, we verify that the n-times enlarged disks $\Delta_i = \Delta(\tilde{z}_i, R_i) = \Delta(\tilde{z}_i, n \cdot r_i)$ are disjoint and that each of them contains exactly the same number of approximations as roots of p counted with multiplicity. As in [19, 52], we use Rouché's theorem for the verification step. If the clustering and the verification succeed, we return the disks $\Delta_1, \ldots, \Delta_k$ and the number of approximations $\hat{z} \in \{\hat{z}_1, \ldots, \hat{z}_n\}$ in the disk as the multiplicity of the root isolated by the disk. If either clustering or verification does not succeed, we repeat with a higher precision. Strzebonski [49] has previously described a similar approach. The main difference is that he used a heuristic for the clustering step and hence could neither prove completeness of his approach nor analyze its complexity. He reports that his algorithm does very well in the context of CAD computation.

In the example above, we would have the additional information that p has exactly two distinct roots. We ask the oracle for an L-approximation of p for sufficiently large L and approximately factor it. Suppose that we obtain approximations -1.01, 1.4, and 1.42 of the roots, and let $\hat{p} = (x+1.01)(x-1.4)(x-1.42)$. The clustering step may then put the first approximation into a singleton cluster and the other two approximations into a cluster of size two. It also computes disjoint enclosing disks. The verification step tries to certify that p and $\hat{p}$ contain the same number of roots in both disks. If L and b are sufficiently large, clustering and verification succeed.

If Pan's algorithm [36] is used for the approximate factorization step, then the overall algorithm has bit complexity[5]

$$\tilde{O}\left(n^3 + n^2 \sum_{i=1}^{k} \log M(z_i) + n \sum_{i=1}^{k} \log\left(M(\sigma_i^{-m_i}) \cdot M(P_i^{-1})\right)\right) \tag{2}$$

where $P_i := \prod_{j \neq i} (z_i - z_j)^{m_j} = \frac{p^{(m_i)}(z_i)}{m_i! p_n}$, and $M(x) := \max(1, |x|)$. Observe that our algorithm is adaptive in a very strong sense, namely, the above bound exclusively depends on the actual multiplicities and the geometry (i.e. the actual modulus of the roots and their distances to each other) of the roots. There is also no dependency on the size or the type (i.e. whether they are rational, algebraic or transcendental) of the coefficients of p.

Our algorithm can also be used to further refine the isolating disks to a size of $2^{-\kappa}$ or less, where κ is a given integer. The bit complexity for the refinement is given by the bound in (2) plus an additional term $\tilde{O}(n \cdot \kappa \cdot \max_i m_i)$. In particular, for square-free polynomials, the amortized cost per root and bit of precision is $\tilde{O}(1)$, and hence the method is optimal up to polylogarithmic factors.

For the benchmark problem of isolating all roots of a polynomial p with *integer* coefficients of absolute value bounded by 2^τ, the bound in (2) becomes $\tilde{O}(n^3 + n^2\tau)$.[6] The bound for the refinement becomes $\tilde{O}(n^3 + n^2\tau + n\kappa)$, even if there exist multiple roots.

For a square-free integer polynomial p, an algorithm by Pan [17, Theorem 3.1] achieves a comparable complexity bound for the benchmark problem. That is, based on the computations

[4] This is crucial to control the cost for the final verification step. For details, we refer to Sections 2.2.2 and 2.2.3.

[5] $\tilde{O}$ indicates that we omit logarithmic factors.

[6] We first divide p by its leading coefficient to meet the requirement on the leading coefficient, and apply our algorithm to p/p_n.

in [46, Section 20], one can compute a bound b_0 of size $\Theta(n(\tau+\log n))$ with the property that if n points $\hat{z}_j \in \mathbb{C}$ fulfill the inequality (1) for a $b \geq b_0$, then they approximate the corresponding roots z_j to an error less than $\sigma_j/(2n)$; cf. Lemma 3 for an adaptive version. Hence, for $b \geq b_0$, Pan's factorization algorithm also yields isolating disks for the roots of p using $\tilde{O}(n^2\tau)$ bit operations. Note while this approach achieves a good worst case complexity, however, it is for the price of running the factorization algorithm with $b = \Theta(n(\tau+\log n))$ even when the roots are well conditioned. In contrast, our algorithm turns Pan's factorization algorithm into a highly adaptive method for isolating and approximating the roots of a general polynomial. Also, for general polynomials, there exist bounds [46, Section 19] for the distance between the roots of p and corresponding approximations fulfilling (1). They are optimal for roots of multiplicity $\Omega(n)$ but overestimate badly if all roots have considerably smaller multiplicities. For root refinement, the bit complexity of our method depends on κ as $\tilde{O}(n \max_i m_i \cdot \kappa)$ and, hence, it adapts to the highest occurring multiplicity, whereas previous methods [28, 36, 43] depend as $\tilde{O}(n^2\kappa)$.

Topology Computation and Computing Real Solutions of Bivariate Systems Our new root isolation algorithms has an interesting consequence on the complexity of computing the topology (in terms of a cylindrical algebraic decomposition) of a real planar algebraic curve specified as the zero set of an integer polynomial and of isolating the real solutions of a bivariate polynomial system. Both problems are well-studied [2, 3, 4, 6, 7, 9, 10, 12, 13, 14, 15, 16, 20, 26, 29, 41, 49]. The latter problem can be reduced to the former as the real solutions of the bivariate system $f(x,y) = g(x,y) = 0$ correspond to the points on the real curve $f^2(x,y) + g^2(x,y) = 0$. In Section 3, we apply our method to a recent algorithm TOPNT [4] for computing the topology of a planar algebraic curve. This yields bounds on the *expected* number of bit operations. which improve the currently best (which are both deterministic) bounds [15, 29] from $\tilde{O}(n^9\tau + n^8\tau^2)$ to $\tilde{O}(n^6 + n^5\tau)$ for topology computation and from $\tilde{O}(n^8 + n^7\tau)$ to $\tilde{O}(n^6 + n^5\tau)$ for solving bivariate systems.

As several other recent algorithms [7, 10, 11, 14, 49] for topology computation or bivariate system solving, TOPNT uses numerical computation as much as possible. In particular, the symbolic operations are restricted to resultant and gcd computations, which do not dominate the overall bit complexity. The workhorse in TOPNT is root isolation and refinement as considered in the first part of this paper, in particular, the isolation of the roots of the "fiber" polynomials $f(\alpha,y) \in \mathbb{C}[y]$, where α is an x-critical point of a planar algebraic curve defined as the vanishing set of a polynomial $f \in \mathbb{Z}[x,y]$. The number of distinct roots of $f(\alpha,y)$ is available from an algebraic precomputation. Combining the adaptive complexity bounds from this paper and the amortized complexity bounds from [29] for all fiber polynomials $f(\alpha,y)$ eventually yields considerably improved complexity bounds for the numerical steps.

Paper History: An extended abstract [34] of this paper was presented at ISSAC 2013. The current paper extends the conference version significantly. In particular, the analysis of the algorithm for root isolation (i.e. the results in Section 2) was only sketched (Lemma 1 and Theorem 1 were stated without proof, and only a sketch of the proof of Theorem 5 was given), and the application of our root isolation algorithm to curve topology computation and to solving bivariate polynomial systems as well as the corresponding analysis (i.e. Section 3) was not covered at all in the extended abstract.

2. Root Finding

2.1. Setting and Basic Properties

We consider a polynomial

$$p(x) = p_n x^n + \ldots + p_0 \in \mathbb{C}[x] \tag{3}$$

of degree $n \geq 2$, where $1/4 \leq p_n \leq 1$. We fix the following notations:

- $M(x) := \max(1, |x|)$, for $x \in \mathbb{R}$,
- τ_p denotes the minimal non-negative integer with $\frac{|p_i|}{|p_n|} \leq 2^{\tau_p}$ for all $i = 0, \ldots, n-1$,
- $\|p\| := \|p\|_1 := |p_0| + \ldots + |p_n|$ denotes the 1-norm of p,
- $z_1, \ldots, z_k \in \mathbb{C}$ are the distinct complex roots of p, with $k \leq n$,
- $m_i := \operatorname{mult}(z_i, p)$ is the multiplicity of z_i,
- $\sigma_i := \sigma(z_i, p) := \min_{j \neq i} |z_i - z_j|$ is the *separation* of z_i,
- $\Gamma_p := M(\max_i \log M(z_i))$ denotes the *logarithmic root bound* of p,
- $\operatorname{Mea}(p) = |p_n| \cdot \prod_i M(z_i)^{m_i}$ denotes the *Mahler Measure* of p.

The quantities τ_p, Γ_p, $|p_n|$ and $\operatorname{Mea}(p)$ are closely related.

Lemma 1. $\Gamma_p \leq 1 + \tau_p$ and $\tau_p - n - 1 \leq \log \frac{\operatorname{Mea}(p)}{|p_n|} \leq n\Gamma_p$.

Proof. By Cauchy's root bound $\max_i |z_i| \leq 1 + \max_i |p_i| / |p_n|$, and thus $\max_i \log |z_i| \leq 1 + \tau_p$. Since $\tau_p \geq 0$, by definition, we have $\Gamma_p \leq 1 + \tau_p$. The i-th coefficient of p is smaller than or equal to $\binom{n}{i} \operatorname{Mea}(p) \leq 2^n \operatorname{Mea}(p) \leq 2^{n(\Gamma_p + 1)}$. Thus, from the definition of τ_p, either $\tau_p = 0$ or $2^n \frac{\operatorname{Mea}(p)}{|p_n|} \geq \max_i \frac{|p_i|}{|p_n|} \geq 2^{\tau_p - 1}$ $\square$

We assume the existence of an oracle which provides arbitrary good approximations of the polynomial p. Let $L \geq 1$ be an integer. We call a polynomial $\tilde{p} = \tilde{p}_n x^n + \ldots + \tilde{p}_0$, with $\tilde{p}_i = s_i \cdot 2^{-(L+1)}$ and $s_i \in \mathbb{Z}$, an *absolute L-approximation* of p if $|\tilde{p}_i - p_i| \leq 2^{-L}$. We further assume that we can ask for such an approximation $\tilde{p}$ for the cost $\tilde{O}(n(L + \tau_p))$. This is the cost of reading the coefficients of $\tilde{p}$.

We call a polynomial $\tilde{p} = \tilde{p}_n x^n + \ldots + \tilde{p}_0$, with $\tilde{p}_i = s_i \cdot 2^{-(L+1)}$ and $s_i \in \mathbb{Z}$, a *relative L-approximation* of p if $\|\tilde{p} - p\| \leq 2^{-L} \|p\|$. Since $L \geq 1$, the triangle inequality implies that

$$\frac{\|\tilde{p}\|}{2} \leq \|p\| \leq 2\|\tilde{p}\|. \tag{4}$$

Furthermore, notice that any absolute $(L + \lceil \log(n+1) \rceil + 2)$-approximation of p is also a relative L-approximation of p because of $\|\tilde{p} - p\| \leq (n+1) \cdot 2^{-L - \lceil \log(n+1) \rceil - 2} \leq |p_n| \cdot 2^{-L} \leq \|p\| \cdot 2^{-L}$. Hence, we can ask for a relative L-approximation for the cost $\tilde{O}(n(L + \tau_p)) = \tilde{O}(n(L + \operatorname{Mea}(p)))$.

In the next step, we show that a "good" integer approximation Γ of Γ_p can be computed with $\tilde{O}(n^2 \Gamma_p)$ bit operations.

Theorem 1. An integer $\Gamma \in \mathbb{N}$ with

$$\Gamma_p \leq \Gamma < 8 \log n + \Gamma_p \tag{5}$$

can be computed with $\tilde{O}(n^2 \Gamma_p)$ bit operations. The computation uses an absolute L-approximation of precision L of p with $L = O(n\Gamma_p)$.

Proof. We consider the *Cauchy polynomial*

$$\bar{p}(x) := |p_n|x^n - \sum_{i=0}^{n-1} |p_i|x^i$$

of p. Then, according to [37, Thm. 8.1.4.] or [48, Thm. 3.8(e)], $\bar{p}$ has a unique positive real root $\xi \in \mathbb{R}^+$, and the following inequality holds:

$$\max_i |z_i| \leq \xi < \frac{n}{\ln 2} \cdot \max_i |z_i| < 2n \cdot \max_i |z_i|.$$

It follows that $\bar{p}(x) > 0$ for all $x \geq \xi$ and $\bar{p}(x) < 0$ for all $x < \xi$. Furthermore, since $\bar{p}$ coincides with its own Cauchy polynomial, each complex root of $\bar{p}$ has absolute value less than or equal to $|\xi|$. Let k_0 be the smallest non-negative integer k with $\bar{p}(2^k) > 0$ (which is equal to the smallest k with $2^k > \xi$). Our goal is to compute an integer Γ with $k_0 \leq \Gamma \leq k_0 + 1$. Namely, if Γ fulfills the latter inequality, then $M(\max_i |z_i|) \leq M(\xi) \leq 2^\Gamma < 4M(\xi) < 8n \cdot M(\max_i |z_i|)$, and thus Γ fulfills inequality (5). In order to compute a Γ with $k_0 \leq \Gamma \leq k_0 + 1$, we use exponential and binary search (try $k = 1, 2, 4, 8, \ldots$ until $\bar{p}(2^k) > 0$ and, then, perform binary search on the interval $k/2$ to k) and approximate evaluation of $\bar{p}$ at the points 2^k: More precisely, we evaluate $\bar{p}(2^k)$ using interval arithmetic with a precision ρ (using fixed point arithmetic) which guarantees that the width w of $\mathfrak{B}(\bar{p}(2^k), \rho)$ is smaller than 1, where $\mathfrak{B}(E, \rho)$ is the interval obtained by evaluating a polynomial expression E via interval arithmetic with precision ρ for the basic arithmetic operations; see [28, Section 4] for details. We use [28, Lemma 3] to estimate the cost for each such evaluation: Since $\bar{p}$ has coefficients of size less than $2^{\tau_p}|p_n| < 2^{\tau_p}$, we have to choose ρ such that

$$2^{-\rho+2}(n+1)^2 2^{\tau_p + nk} < \frac{1}{4}$$

in order to ensure that $w < 1/4$. Hence, ρ is bounded by $O(\tau_p + nk)$ and, thus, each interval evaluation needs $\tilde{O}(n(\tau_p + nk))$ bit operations. We now use exponential plus binary search to find the smallest k such that $\mathfrak{B}(\bar{p}(2^k), \rho)$ contains only positive values. The following argument then shows that $k_0 \leq k \leq k_0 + 1$: Obviously, we must have $k \geq k_0$ since $\bar{p}(2^k) < 0$ and $\bar{p}(2^k) \in \mathfrak{B}(\bar{p}(2^k), \rho)$ for all $k < k_0$. Furthermore, the point $x = 2^{k_0+1}$ has distance more than 1 to each of the roots of $\bar{p}$, and thus $|\bar{p}(2^{k_0+1})| \geq |p_n| \geq 1/4$. Hence, it follows that $\mathfrak{B}(\bar{p}(2^{k_0} + 1), \rho)$ contains only positive values. For the search, we need

$$O(\log k_0) = O(\log \log \xi) = O(\log(\log n + \Gamma_p))$$

iterations, and the cost for each of these iterations is bounded by $\tilde{O}(n(\tau_p + nk_0)) = \tilde{O}(n^2 \Gamma_p)$ bit operations. □

2.2. *Algorithm*

We present an algorithm for isolating the roots of a polynomial $p(x) = \sum_{i=0}^n p_i x^i = p_n \prod_{i=1}^k (x - z_i)^{m_i}$, where the coefficients p_i are given as described in the previous section. We may assume that $k > 1$; the problem is trivial otherwise. If $k = 1$, then $-p_{n-1}/(np_n)$ is the root of multiplicity n. The algorithm uses some polynomial factorization algorithm to produce approximations for the roots $z_1, \ldots, z_k$, and then performs a clustering and certification step to verify that the candidates are of high enough quality. For concreteness, we pick Pan's factorization algorithm [36] for the factorization step, which also currently offers the best worst case bit complexity.[7] If the candidates do not pass the verification step, we reapply the factorization algorithm with a higher precision. Given a polynomial p with $|z_i| \leq 1$ for $1 \leq i \leq k$, and a positive integer b denoting the desired precision, the factorization algorithm computes n root approximations $\hat{z}_1, \dots, \hat{z}_n$. The quality of approximation and the bit complexity are as follows:

Theorem 2 (Pan [36])**.** Suppose that $|z_i| \leq 1$ for $1 \leq i \leq k$. For any positive integer $b \geq n \log n$, complex numbers $\hat{z}_1, \dots, \hat{z}_n$ can be computed such that they satisfy

$$\left\|p - p_n \prod_{i=1}^n (x - \hat{z}_i)\right\| \leq 2^{-b} \|p\|$$

using $\tilde{O}(n)$ operations performed with the precision of $O(b)$ bits (or $\tilde{O}(bn)$ bit-operations). The input to the algorithm is a relative L-approximation of p, where $L = O(b)$. We write $\hat{p} := p_n \prod_{i=1}^n (x - \hat{z}_i)$. The algorithm returns the real and imaginary part of the $\hat{z}_i$'s as dyadic fractions of the form $A \cdot 2^{-B}$ with $A \in \mathbb{Z}$, $B \in \mathbb{N}$ and $B = O(b)$. All fractions have the same denominator.

The parameter b controls the quality of the resulting approximations. Note that Pan's algorithm requires all roots of the input polynomial to lie within the unit disk $\Delta(0,1)$. Hence, in order to apply the above result to our input polynomial, we first scale p such that the roots come to lie in the unit disk. That is, we compute a Γ as in Theorem 1, and then consider the polynomial $f(x) := p(s \cdot x) = \sum_{i=0}^n f_i x^i$ with $s := 2^\Gamma$. Then, $f(x)$ has roots $\xi_i = z_i/s \in \Delta(0,1)$, and thus we can use Pan's Algorithm with $b' := n\Gamma + b$ to compute an approximate factorization $\hat{f}(x) := \sum_{i=0}^n \hat{f}_i x^i := f_n \prod_{i=1}^n (x - \hat{\xi}_i)$ such that $\left\|f - \hat{f}\right\| < 2^{-b'} \|f\|$. Let $\hat{z}_i := s \cdot \hat{\xi}_i$ for all i and $\hat{p}(x) := p_n \cdot \prod_{i=1}^n (x - \hat{z}_i) = \hat{f}(x/s) = \sum_{i=0}^n \hat{f}_i / s^i x^i$, then

$$\|\hat{p} - p\| = \sum_{i=0}^n |f_i/s^i - \hat{f}_i/s^i| \leq \sum_{i=0}^n |f_i - \hat{f}_i| \leq 2^{-b'} \sum_{i=0}^n |f_i| \leq 2^{-b'} s^n \sum_{i=0}^n |f_i/s^i| = 2^{-b} \|p\|.$$

For the factorization of f, we need a relative b'-approximation of f, and thus a relative L-approximation of p with $L = O(n\Gamma + b) = \tilde{O}(n\Gamma_p + b)$. The total cost is $\tilde{O}(n^2\Gamma_p + nb)$ bit operations. We summarize in:

Corollary 1. For an arbitrary polynomial $p = p_n \cdot x^n + \cdots + p_0 \in \mathbb{C}[x]$, with $1/4 \leq |p_n| \leq 1$, and an integer $b \geq n \log n$, complex numbers $\hat{z}_1, \dots, \hat{z}_n$ can be computed such that

$$\left\|p - p_n \prod_{i=1}^n (x - \hat{z}_i)\right\| \leq 2^{-b} \|p\|$$

using $\tilde{O}(n^2\Gamma_p + bn)$ bit-operations. We write $\hat{p} := p_n \prod_{i=1}^n (x - \hat{z}_i)$. The algorithm returns the real and imaginary part of the $\hat{z}_i$'s as dyadic fractions of the form $A \cdot 2^{-B}$ with $A \in \mathbb{Z}$, $B \in \mathbb{N}$ and $B = O(b + n\Gamma_p)$. All fractions have the same denominator.

We now examine how far the approximations $\hat{z}_1, \dots, \hat{z}_n$ can deviate from the actual roots for a given value of b, i.e., a quantitative version of the fact that the roots of a polynomial depend

[7] In practice, one might consider a numerical root finder [5] based on the Aberth-Ehrlich method instead. There is empirical evidence that such methods achieve comparable complexity bounds. We further remark that many solvers only provide approximations $\hat{z}_1, \dots, \hat{z}_n$ of the roots without any guarantee on the error $\|p - p_n \cdot \prod_i (x - \hat{z}_i)\|$. In this case, we first have to estimate the latter error by an algorithm for approximate polynomial multiplication; e.g. the method from [45] allows us to approximate the product $p_n \cdot \prod_i (x - \hat{z}_i)$ to an absolute error of 2^{-b} using $\tilde{O}(n(n\Gamma_p + b))$ bit operations. Obviously, if $\hat{z}_i \to z_i$ for all i, then $\|p - p_n \cdot \prod_i (x - \hat{z}_i)\| \to 0$, hence we can alternatively assume that our oracle provides arbitrary good approximations of the roots (without any additional estimate on the actual error).

continuously on the coefficients. Such estimates are well known, e.g., [47, Theorem 2.7] and [24, Theorem 4.10c]. For our complexity bounds, we also need the dependency on the multiplicities and the root separation and hence need to state our own bounds. Technically, there is nothing new here. Let $\Delta(z,r)$ be the disk with center z and radius r and let $\operatorname{bd}\Delta(z,r)$ be its boundary. We further define $P_i := \prod_{j\neq i}(z_i - z_j)^{m_j}$. Then, $p^{(m_i)}(z_i) = m_i! p_n P_i$.

Lemma 2. If $r \leq \sigma_i/n$, then

$$|p(x)| > \frac{r^{m_i} \cdot |p_n P_i|}{4}$$

for all x on the boundary of $\Delta(z_i, r)$.

Proof. We have

$$\begin{aligned}|p(x)| &= |p_n| \cdot |x - z_i|^{m_i} \cdot \prod_{j\neq i} |x - z_j|^{m_j} \geq |p_n| \cdot |x - z_i|^{m_i} \cdot \prod_{j\neq i} |z_i - z_j|^{m_j} \cdot (1 - |x - z_i|/|z_i - z_j|)^{m_j} \\ &\geq r^{m_i}(1 - 1/n)^{n-m_i} |p_n| \cdot \prod_{j\neq i} |z_i - z_j|^{m_j} > r^{m_i} \cdot |p_n P_i|/4.\end{aligned}$$

□

Based on the above Lemma, we can now use Rouché's theorem [8] to show that, for sufficiently large b, the disk $\Delta(z_i, 2^{-b/(2m_i)})$ contains exactly m_i root approximations.

Lemma 3. Let $\hat{p}$ be such that $\|p - \hat{p}\| \leq 2^{-b}\|p\|$. If

$$b \geq \max(8n, n\log(n)), \text{ and } b \text{ is a power of two} \tag{6}$$

$$2^{-b/(2m_i)} \leq \frac{1}{2n^2}, \tag{7}$$

$$2^{-b/(2m_i)} \leq \frac{\sigma_i}{2n}, \text{ and} \tag{8}$$

$$2^{-b/2} \leq \frac{|P_i|}{16(n+1)2^{\tau_p} M(z_i)^n} \tag{9}$$

for all i, the disk $\Delta(z_i, 2^{-b/(2m_i)})$ contains exactly m_i root approximations. For $i \neq j$, let $\hat{z}_i$ and $\hat{z}_j$ be arbitrary approximations in the disks $\Delta(z_i, 2^{-b/(2m_i)})$ and $\Delta(z_j, 2^{-b/(2m_j)})$, respectively. Then,

$$\left(1 - \frac{1}{n}\right) \cdot |z_i - z_j| \leq |\hat{z}_i - \hat{z}_j| \leq \left(1 + \frac{1}{n}\right) \cdot |z_i - z_j|.$$

Proof. Let

$$\delta_i := \left(16 \cdot (n+1) \cdot 2^{-b} 2^{\tau_p} |P_i|^{-1} M(z_i)^n\right)^{1/m_i}.$$

It is easy to verify that $\delta_i \leq 2^{-b/(2m_i)} \leq \min(1, \sigma_i)/(2n)$. The first inequality follows from (9) and the second inequality follows from (7) and (8). We will show that $\Delta(z_i, \delta_i)$ contains m_i approximations. To this end, is suffices to show that $|(p - \hat{p})(x)| < |p(x)|$ for all x on the boundary of $\Delta(z_i, \delta_i)$. Then, Rouché's theorem guarantees that $\Delta(z_i, \delta_i)$ contains the same number of roots of p and $\hat{p}$ counted with multiplicity. Since z_i is of multiplicity m_i and $\delta_i < \sigma_i/n$, the disk contains

[8] Rouché's theorem states that if f and g are holomorphic functions with $|(f - g)(x)| < |f(x)|$ for all points x on the boundary of some disk Δ, then f and g have the same number of zeros (counted with multiplicity) in Δ.

exactly m_i roots of p counted with multiplicity. We have (note that $|x| \leq (1+1/(2n^2)) \cdot M(z_i)$ for $x \in \operatorname{bd}\Delta(z_i,\delta_i)$)

$$\begin{aligned}
|(p-\hat{p})(x)| &\leq \|p-\hat{p}\| \cdot M(x)^n < 2^{-b}\,\|p\|\,M(x)^n \\
&\leq 2^{-b}\,\|p\| \cdot (1+1/(2n^2))^n \cdot M(z_i)^n \\
&\leq 4 \cdot 2^{-b} \cdot 2^{\tau_p}|p_n| \cdot (n+1) \cdot M(z_i)^n \\
&\leq \delta_i^{m_i}|p_n P_i|/4 < |p(x)|,
\end{aligned}$$

where the inequality in line three follows from $\|p\| \leq (n+1)|p_n|2^{\tau_p}$, the first one in line four follows from the definition of δ_i, and the last inequality follows from Lemma 2. It follows that $\Delta(z_i, 2^{-b/(2m_i)})$ contains exactly m_i approximations. Furthermore, since $\delta_i \leq \sigma_i/(2n)$ for all i, the disks $\Delta(z_i,\delta_i)$, $1 \leq i \leq k$, are pairwise disjoint.

For the second claim, we observe that $|\hat{z}_\ell - z_\ell| \leq 2^{-b/(2m_\ell)} \leq \sigma_\ell/(2n) \leq |z_i - z_j|/(2n)$ for $\ell = i,j$ and hence $|\hat{z}_i - z_i| + |\hat{z}_j - z_j| \leq |z_i - z_j|/n$. The claim now follows from the triangle inequality. □

We have now established that the disks $\Delta(z_i, 2^{-b/(2m_i)})$, $1 \leq i \leq k$, are pairwise disjoint and that the i-th disk contains exactly m_i root approximations provided that b satisfies (6) to (9). We want to stress that the radii $2^{-b/(2m_i)}$, $1 \leq i \leq k$, are vastly different. For example, assume $b = 40$. For a one-fold root ($m = 1$), the radius is 2^{-20}, for a double root ($m = 2$) the radius is 2^{-10}, for a four-fold root ($m = 4$) the radius is 2^{-5}, and for a twenty-fold root ($m = 20$), the radius is as large as $1/2$. Unfortunately, the conditions on b are stated in terms of the quantities m_i, σ_i and $|P_i|$ which we do not know. Also, we do not know the center z_i. In the remainder of the section, we will show how to cluster root approximations and to certify them. We will need the following more stringent properties for the clustering and certification step.

$$2^{-b/(2m_i)} < \min\left(\left(\frac{\sigma_i}{4n}\right)^8, \frac{\sigma_i}{1024n^2}\right) \tag{10}$$

$$2^{-b/8} < \min\left(\frac{1}{16}, \frac{|P_i|}{(n+1) \cdot 2^{2n\Gamma_p + 8n}}\right) \tag{11}$$

Let b_0 be the smallest integer satisfying (6) to (11) for all i. Then,

$$b_0 = O(n\log n + n\Gamma_p + \max_i(m_i \log M(\sigma_i^{-1}) + \log M(P_i^{-1}))).$$

We next provide a high-level description of our algorithm to isolate the roots of p. The details of the clustering step and the certification step are then given in Sections 2.2.2 and 2.2.3, respectively.

2.2.1. *Overview of the Algorithm*

On input p and the number k of distinct roots, the algorithm outputs isolating disks $\Delta_i = \Delta(\tilde{z}_i, R_i)$ for the roots of p as well as the corresponding multiplicities m_i. The radii satisfy $R_i < \sigma_i/(64n)$.

The algorithm uses the factorization step with an increasing precision until the result can be certified. If either the clustering step or the certification step fails, we simply double the precision. There are a couple of technical safeguards to ensure that we do not waste time on iterations with an insufficiently large precision (Steps 2, 5, and 6); also recall that we need to scale our initial polynomial.

(1) Compute the bound 2^{Γ} for the modulus of all roots of p, where Γ fulfills Inequality (5). According to Theorem 1, this can be done with $\tilde{O}(n^2\Gamma_p)$ bit operations.
(2) Compute a 2-approximation $\lambda = 2^{l_\lambda}$, with $l_\lambda \in \mathbb{Z}$, of $\|p\|/|p_n|$. According to (4), this computation needs $\tilde{O}(n\tau_p) = \tilde{O}(n^2\Gamma_p)$ bit operations.
(3) Scale p, that is, $f(x) := p(s\cdot x)$, with $s := 2^{\Gamma}$, to ensure that the roots $\xi_i = z_i/S$, $i = 1,\dots,k$, of f are contained in the unit disk. Let b be the smallest integer satisfying (6)
(4) Run Pan's algorithm on input f with parameter $b' := b + n\Gamma$ to produce approximations $\hat{\xi}_1,\dots,\hat{\xi}_n$ for the roots of f. Then, $\hat{z}_i := s\cdot\hat{\xi}_i$ are approximations of the roots of p, and $\|\hat{p}-p\| < 2^{-b}\|p\|$, where $\hat{p}(x) := p_n\prod_{i=1}^n(x-\hat{z}_i)$.
(5) If there exists a $\hat{z}_i$ with $\hat{z}_i \geq 2^{\Gamma+1}$, return to Step 4 with $b := 2b$.
(6) If $\prod_{i=1}^n M(\hat{z}_i) > 8\lambda$, return to Step 4 with $b := 2b$.
(7) Partition $\hat{z}_1,\dots,\hat{z}_n$ into k clusters $C_1,\dots,C_k$. Compute (well separated) enclosing disks $D_1,\dots,D_k$ for the clusters. For details, see Section 2.2.2. If the clustering fails to find k clusters and corresponding disks, return to Step 4 with $b := 2b$.
(8) For each i, let Δ_i denote the disk with the same center as D_i but with an n-times larger radius. We now verify the existence of $|C_i|$ roots (counted with multiplicity) of p in Δ_i. For details of the verification, see Section 2.2.3. If the verification fails, return to Step 4 with $b := 2b$.
(9) If the verification succeeds, output the disks Δ_i (in Step 7, we guarantee that the disks Δ_i are pairwise disjoint) and report the number $|C_i|$ of root approximations $\hat{z} \in \{\hat{z}_1,\dots,\hat{z}_n\}$ contained in the disks as the corresponding multiplicities.

Notice that Steps 5 and 6 ensure that $\log M(\hat{z}_i) = O(\Gamma_p + \log n)$ for all i, and that $\log\prod_{i=1}^n M(\hat{z}_i) = O(\log(\|p\|/|p_n|)) = O(\log n + \tau_p) = \tilde{O}(n\Gamma_p)$. The following Lemma guarantees that the algorithm passes these steps if $b \geq b_0$.

Lemma 4. For any $b \geq b_0$, it holds that $|\hat{z}_i| < 2^{\Gamma+1}$ for all i, and $\prod_{i=1}^n M(\hat{z}_i) < 8\lambda$.

Proof. In the proof of Lemma 3, we have already shown that $|\hat{z}_i| \leq (1+1/(2n^2))\cdot M(z_i)$ for all i. Hence, it follows that $|\hat{z}_i| \leq (1+1/(2n^2))\cdot 2^{\Gamma_p} < 2\cdot 2^{\Gamma_p} \leq 2^{\Gamma_p+1}$, and

$$\prod_{i=1}^n M(\hat{z}_i) \leq 4\cdot\prod_{i=1}^k M(z_i)^{m_i} < \frac{4\,\mathrm{Mea}(p)}{|p_n|} \leq \frac{4\|p\|_2}{|p_n|} \leq \frac{4\|p\|}{|p_n|} < 8\lambda.$$

□

2.2.2. Clustering

After candidate approximations $\hat{z}_1,\dots,\hat{z}_n$ are computed using a fixed precision parameter b, we perform a partitioning of these approximations into k clusters $C_1,\dots,C_k$, where k is given as an input. The clustering is described in detail below. It works in phases. At the beginning of a phase, it chooses an unclustered approximation and uses it as the seed for the cluster formed in this phase. Ideally, each of the clusters corresponds to a distinct root of p. The clustering algorithm satisfies the following properties:
(1) For $b < b_0$, the algorithm may or may not succeed in finding k clusters.
(2) For $b \geq b_0$, the clustering always succeeds.

Whenever the clustering succeeds, the cluster C_i with seed $\tilde{z}_i$ is contained in the disk $D_i := \Delta(\tilde{z}_i, r_i)$, where $r_i \approx \min(\frac{1}{n^2}, \frac{\tilde{\sigma}_i}{256n^2})$, and $\tilde{\sigma}_i = \min_{j\neq i}|\tilde{z}_i - \tilde{z}_j|$. Furthermore, for $b \geq b_0$, D_i contains the root z_i (under suitable numbering) and exactly m_i many approximations.

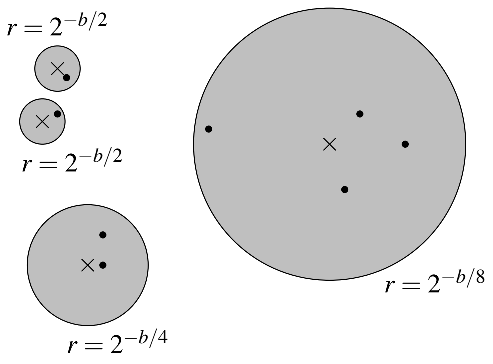

Fig. 1. Example of a polynomial with four distinct roots with multiplicities 1, 1, 2, and 4. Crosses are roots of the polynomial, dots represent the approximations. The disk around a root shows the potential locations of its approximations. Note that the straight-forward approach to cluster with a fixed distance threshold fails for all b with $b < (\max_i m_i) \cdot \log(\min_i \sigma_i)^{-1}$: For each such b, one can not choose any threshold that allows detecting the simple roots without splitting the four-fold root.

Before we describe our clustering method, we discuss two evident approaches that do not work for any b of size comparable to b_0 or smaller. A clustering with a fixed grid does not work as root approximations coming from roots with different multiplicities may move by vastly distinct amounts. As a consequence, we can only succeed if $b > (\max_i m_i) \cdot \log(\min_i \sigma_i)^{-1}$ which can be considerably larger than b_0, see Figure 1. A clustering based on Gershgorin disks does not work either because very good approximations of a multiple root lead to large disks which then fail to separate approximations of distinct roots. In particular, if approximations are identical, the corresponding Gershgorin disks have infinite radius.

For our clustering, we use the fact that the factorization algorithm provides approximations $\hat{z}$ of the root z_i with distance less than $2^{-b/(2m_i)}$ (for $b \geq b_0$). Thus, we aim to determine clusters C of maximal size such that the pairwise distance between two elements in the same cluster is less than $2 \cdot 2^{-b/(2|C|)}$. We give details.

(1) Initialize $\mathscr{C}$ to the empty set (of clusters).
(2) Initialize C to the set of all unclustered approximations and choose $\hat{z} \in C$ arbitrarily. Let $a := 2^{\lfloor \log n \rfloor + 2}$ and $\delta := 2^{-b/4}$.
(3) Update C to the set of points $q \in C$ satisfying $|\hat{z} - q| \leq 2 \sqrt[a/2]{\delta}$.
(4) If $|C| \geq a/2$, add C to $\mathscr{C}$. Otherwise, set $a := a/2$ and continue with step 3.
(5) If there are still unclustered approximations, continue with step 2.
(6) If the number of clusters in $\mathscr{C}$ is different from k, report failure, double b and go back to the factorization step.

Note that, for $b \geq b_0$, the disks $\Delta(z_i, 2^{-b/(2m_i)})$ are disjoint. Let Z_i denote the set of root approximations in $\Delta(z_i, 2^{-b/(2m_i)})$. Then, $|Z_i| = m_i$ according to Lemma 3. We show that, for $b \geq b_0$, the clustering algorithm terminates with $C = Z_i$ if called with an approximation $\hat{z} \in Z_i$.

Lemma 5. Assume $b \geq b_0$, $\hat{z}_i \in Z_i$, $\hat{z}_j \in Z_j$, and $i \neq j$. Then,

$$\left|\hat{z}_i - \hat{z}_j\right| \geq 2 \cdot (2^{-b/(16m_i)} + 2^{-b/(16m_j)}).$$

Proof. Since $b \geq b_0$, we have $2^{-b/(2m_\ell)} \leq \sigma_\ell$ for $\ell = i,j$ by (8) and $2^{-b/(16m_\ell)} = (2^{-b/(2m_\ell)})^{1/8} \leq \sigma_\ell/(4n) \leq \sigma_\ell/8$ by (10). Thus,

$$\left|\hat{z}_i - \hat{z}_j\right| \geq \max(\sigma_i,\sigma_j) - 2^{-b/(2m_i)} - 2^{-b/(2m_j)} \geq \frac{\sigma_i}{2} + \frac{\sigma_j}{2} - \frac{\sigma_i}{4} - \frac{\sigma_j}{4} \geq 2\cdot(2^{-b/(16m_i)} + 2^{-b/(16m_j)}).$$

□

Lemma 6. If $b \geq b_0$, the clustering algorithm computes the correct clustering, that is, it produces clusters C_1 to C_k such that $C_i = Z_i$ for all i (under suitable numbering). Let $\tilde{z}_i$ be the seed of C_i and let $\tilde{\sigma}_i = \min_{j\neq i}\left|\tilde{z}_i - \tilde{z}_j\right|$. Then, $(1-1/n)\sigma_i \leq \tilde{\sigma}_i \leq (1+1/n)\sigma_i$ and C_i as well as the root z_i is contained in $\Delta(\tilde{z}_i, \min(\frac{1}{n^2}, \frac{\tilde{\sigma}_i}{256n^2}))$.

Proof. Assume that the algorithm has already produced Z_1 to Z_{i-1} and is now run with a seed $\hat{z} \in Z_i$. We prove that it terminates with $C = Z_i$. Let ℓ be a power of two such that $\ell \leq m_i < 2\ell$. The proof that the algorithm terminates with $C = Z_i$ consists of two parts. We first assume that steps 2 and 3 are executed for $a = 2\ell$. We show that the algorithm will then terminate with $C = Z_i$. In the second part of the proof, we show that the algorithm does not terminate as long as $a > 2\ell$.

Assume the algorithm reaches steps 2 and 3 with $a/2 = \ell$, i.e. $a/2 \leq m_i < a$. For any approximation $q \in Z_i$, we have $|\hat{z} - q| \leq 2\cdot 2^{-b/(2m_i)} = 2\sqrt[m_i/2]{\delta} \leq 2\sqrt[a/2]{\delta}$. Thus, $Z_i \subseteq C$. Conversely, consider any approximation $q \notin Z_i$. Then, $|\hat{z} - q| \geq 2\cdot 2^{-b/(16m_i)} > 2\sqrt[4m_i]{\delta} \geq 2\sqrt[2a]{\delta}$, and thus no such approximation is contained in C. This shows that $C = Z_i$. Since $|C| \geq a/2$, the algorithm terminates and returns Z_i.

It is left to argue that the algorithm does not terminate before $a/2 = \ell$. Since ℓ and a are powers of two, assume we terminate with $a/2 \geq 2\ell$, and let C be the cluster returned. Then, $m_i < a/2 \leq |C| < a$ and Z_i is a proper subset of C. Consider any approximation $q \in C\setminus Z_i$, say $q \in Z_j$ with $j \neq i$. Since $q \notin Z_i$, we have $|q - \hat{z}| \geq 2\cdot(2^{-b/(16m_i)} + 2^{-b/(16m_j)}) > 2\cdot 2^{-b/(16m_i)} > 2\sqrt[4m_i]{\delta}$. And since $q \in C$, we have $|q - \hat{z}| \leq 2\sqrt[a/2]{\delta}$. Thus, $4m_i \leq a/2$ and, hence, there are at least $3a/8$ many approximations in $C\setminus Z_i$. Furthermore, $\left|z_i - z_j\right| \leq |z_i - \hat{z}| + |\hat{z} - q| + \left|q - z_j\right| \leq 2^{-b/(2m_i)} + 2\sqrt[a/2]{\delta} + 2^{-b/(2m_j)} \leq 2^{-b/(16m_i)} + 2\sqrt[a/2]{\delta} + 2^{-b/(16m_j)} \leq 3\sqrt[a/2]{\delta}$. Consequently, there are at least $3a/8$ roots $z_j \neq z_i$ counted with multiplicity within distance $3\sqrt[a/2]{\delta}$ to z_i. This observation allows us to upper bound the value of $|P_i|$, namely

$$|P_i| = \prod_{j\neq i}|z_i - z_j|^{m_j} \leq (3\sqrt[a/2]{\delta})^{3a/8} 2^{(n-m_i-3a/8)\Gamma_p} < 3^n\delta^{3/4}2^{n\Gamma_p} \leq 3^n 2^{-3b/16}2^{n\Gamma_p} < 3^n 2^{-b/8}\cdot 2^{n\Gamma_p},$$

a contradiction to (11).

We now come to the claims about $\tilde{\sigma}_i$ and the disks defined in terms of it. The relation between σ_i and $\tilde{\sigma}_i$ follows from the second part of Lemma 3. All points in $C_i = Z_i$ have distance at most $2\cdot 2^{-b/(2m_i)}$ from $\tilde{z}_i$. Also, by (7) and (10),

$$2\cdot 2^{-b/(2m_i)} < \min(1/n^2, \sigma_i/(512n^2)) \leq \min(1/n^2, \tilde{\sigma}_i/(256n^2))$$

Hence, C_i as well as z_i is contained in $\Delta(\tilde{z}_i, \min(1/n^2, \tilde{\sigma}_i/(256n^2)))$. □

Lemma 7. For a fixed precision b, computing a complete clustering needs $\tilde{O}(nb + n^2\Gamma_p)$ bit operations.

Proof. For each approximation, we examine the number of distance computations we need to perform. Recall that b (property (6)) and a are powers of two, $a \leq 4n$ by definition, and $b \geq 8n \geq 2a$ by property (6). Then, $\sqrt[a/2]{\delta} = 2^{-b/(2a)} \in 2^{-\mathbb{N}}$. Thus, the number $\sqrt[a/2]{\delta}$ has a very simple format in binary notation. There is a single one, and this one is $b/(2a)$ positions after the binary point. In addition, all approximations $\hat{z}$ have absolute value less than $2 \cdot 2^{\Gamma}$ due to Step 5 in the overall algorithm. Thus, each evaluation of the form $|\hat{z} - q| \leq 2 \sqrt[a/2]{\delta}$ can be done with

$$O(\Gamma + \log \delta^{-2/a}) = O((b/a) + \Gamma) = O((b/a) + \Gamma_p + \log n)$$

bit operations.

For a fixed seed $\hat{z}$, in the i-th iteration of step 2, we have at most $a \leq n/2^{i-2}$ many unclustered approximations left in C, since otherwise we would have terminated in an earlier iteration. Hence, we perform at most a evaluations of the form $|\hat{z} - q| \leq 2 \sqrt[a/2]{\delta}$, resulting in an overall number of bit operations of $a \cdot O((b/a) + \Gamma) = O(b + a\Gamma)$ for a fixed iteration. As we halve a in each iteration, we have at most $\log n + 2$ iterations for a fixed $\hat{z}$, leading to a bit complexity of $O(b \log n + n\Gamma) = \tilde{O}(b + n\Gamma) = \tilde{O}(b + n\Gamma_p)$.

In total, performing a complete clustering has a bit complexity of at most $\tilde{O}(nb + n^2\Gamma_p)$. $\square$

When the clustering succeeds, we have k clusters C_1 to C_k and corresponding seeds $\tilde{z}_1, \ldots, \tilde{z}_k \subseteq \{\hat{z}_1, \ldots, \hat{z}_n\}$. For $i = 1, \ldots, k$, we define $D_i := \Delta(\tilde{z}_i, r_i)$, where $\tilde{z}_i$ is the seed for the cluster C_i and

$$r_i := \min(2^{-\lceil 2 \log n \rceil}, 2^{\lceil \log \tilde{\sigma}_i/(256n^2) \rceil}) \geq \min\left(\frac{1}{2n^2}, \frac{\tilde{\sigma}_i}{256n^2}\right). \tag{12}$$

In particular, r_i is a 2-approximation of $\min(1/n^2, \tilde{\sigma}_i/(256n^2))$. Notice that the cost for computing the separations $\tilde{\sigma}_i$ is bounded by $\tilde{O}(nb + n^2\Gamma_p)$ bit operations since we can compute the nearest neighbor graph of the points $\tilde{z}_i$ (and thus the values $\tilde{\sigma}_i$) in $O(n \log n)$ steps [18] with a precision of $O(b + n\Gamma)$.

Now, suppose that $b \geq b_0$, Then, according to Lemma 6, the cluster C_i is contained in the disk D_i. Furthermore, D_i contains exactly one root z_i of p (under suitable numbering of the roots), and it holds that $m_i = \operatorname{mult}(z_i, p) = |C_i|$ and $\min(1/(2n^2), \sigma_i/(512n^2)) \leq r_i \leq \min(1/n^2, \sigma_i/(64n^2))$. If the clustering succeeds for a $b < b_0$, we have no guarantees (actually, the termination condition in step 4 gives some guarantee, however, we have chosen not to exploit it). Hence, before we proceed, we verify that each disk D_i actually contains the cluster C_i. If this is not the case, then we report a failure, return to the factorization step with $b = 2b$, and compute a new corresponding clustering.

In the next and final step, we aim to show that each of the enlarged disks $\Delta_i := \Delta(\tilde{z}_i, R_i) := \Delta(\tilde{z}_i, nr_i)$, $i = 1, \ldots, k$, contains exactly one root z_i of p, and that the number of elements in $C_i \subseteq \Delta_i$ equals the multiplicity of z_i. Notice that, from the definition of r_i and Δ_i, it obvious that the disks Δ_i are pairwise disjoint and that $C_i \subseteq D_i \subseteq \Delta_i$.

2.2.3. *Certification*

In order to show that Δ_i contains exactly one root of p with multiplicity $|C_i|$, we show that each Δ_i contains the same number of roots of p and $\hat{p}$ counted with multiplicity. For the latter, we compute a lower bound for $|\hat{p}(z)|$ on the boundary $\operatorname{bd} \Delta_i$ of Δ_i, and check whether this bound is larger than $|(\hat{p} - p)(z)|$ for all points $z \in \operatorname{bd} \Delta_i$. If this is the case, then we are done according to Rouché's theorem. Otherwise, we start over the factorization algorithm with $b = 2b$. We now come to the details:

(1) Let $\lambda = 2^{l_\lambda}$ be the 2-approximation of $\|p\|/|p_n|$ as defined in step 2 of the overall algorithm.
(2) For $i = 1, \ldots, k$, let $z_i^* := \tilde{z}_i + n \cdot r_i \in \Delta_i$. Note that $|z_i^*| \leq (1 + 1/n) \cdot M(\tilde{z}_i)$ since $nr_i \leq 1/n$.

(3) We try to establish the inequality

$$|\hat{p}(z_i^*)/p_n| > E_i := 64 \cdot 2^{-b} \lambda M(\tilde{z}_i)^n \tag{13}$$

for all i. We will see in the proof of Lemma 9 that this implies that each disk Δ_i contains exactly one root z_i of p and that its multiplicity equals the number $|C_i|$ of approximations within Δ_i. In order to establish the inequality, we consider $\rho = 1, 2, 4, 8, \ldots$ and compute $|\hat{p}(z_i^*)/p_n|$ to an absolute error less than $2^{-\rho}$. If, for all $\rho \le b$, we fail to show that $|p(z_i^*)/p_n| > E_i$, we report a failure and go back to the factorization algorithm with $b = 2b$. Otherwise, let ρ_i be the smallest ρ for which we are successful.

(4) If, at any stage of the algorithm, $\sum_i \rho_i > b$, we also report a failure and go back to the factorization algorithm with $b = 2b$. Lemma 8 then shows that, for fixed b, the number of bit operations that are used for all evaluations is bounded by $\tilde{O}(nb + n^2\tau_p + n^3)$.

(5) If we can verify that $|\hat{p}(\tilde{z}_i + nr_i)/p_n| > E_i$ for all i, we return the disks Δ_i and the multiplicities $m_i = |C_i|$.

Lemma 8. For any i, we can compute $|\hat{p}(z_i^*)/p_n|$ to an absolute error less than $2^{-\rho}$ with a number of bit operations less than

$$\tilde{O}(n(n + \rho + n\log M(\tilde{z}_i) + \tau_p)).$$

For a fixed b, the total cost for all evaluations in the above certification step is upper bounded by $\tilde{O}(nb + n^2\tau_p + n^3)$.

Proof. Consider an arbitrary subset $S \subseteq \{\hat{z}_1, \ldots, \hat{z}_n\}$. We first derive an upper bound for $\prod_{\hat{z}\in S} |z_i^* - \hat{z}|$. For that, consider the polynomial $\hat{p}_S(x) := \prod_{\hat{z}\in S}(x - \hat{z})$. The i-th coefficient of $\hat{p}_S$ is bounded by $\binom{|S|}{i} \cdot \prod_{\hat{z}\in S} M(\hat{z}) \le 2^n \prod_{i=1}^n M(\hat{z}_i) \le 8\lambda \cdot 2^n$ due to step 6 in the overall algorithm. It follows that

$$\prod_{\hat{z}\in S} |z_i^* - \hat{z}| = |\hat{p}_S(z_i^*)| \le (n+1)M(z_i^*)^n \cdot 8\lambda \cdot 2^n < 64(n+1)^2 \cdot 2^n 2^{\tau_p} M(\tilde{z}_i)^n$$

In order to evaluate $|\hat{p}(z_i^*)/p_n| = \prod_{j=1}^n |z_i^* - \hat{z}_j|$, we use approximate interval evaluation with an absolute precision $K = 1, 2, 4, 8, \ldots$. More precisely, we compute the distance of z_i^* to each of the points $\hat{z}_j$, $j = 1, \ldots, n$, up to an absolute error of 2^{-K}, and then take the product over all distances using a fixed point precision of K bits after the binary point. [9] We stop when the resulting interval has size less than $2^{-\rho}$. The above consideration shows that all intermediate results have at most $O(n + \tau_p + n\log M(\tilde{z}_i)$ bits before the binary point. Thus, we eventually succeed for an $K = O(\rho + \tau_p + n + n\log M(\tilde{z}_i))$. Since we have to perform n subtractions and n multiplications, the cost is bounded by $\tilde{O}(nK)$ bit operations for each K. Hence, the bound for the evaluation of $|\hat{p}(z_i^*)/p_n|$ follows.

We now come to the second claim. Since we double ρ in each iteration and consider at most $\log b$ iterations, the cost for the evaluation of $|\hat{p}(z_i^*)/p_n|$ are bounded by $\tilde{O}(n(n + \rho_i + n\log M(\tilde{z}_i) + \tau_p))$. Since we ensure that $\sum_i \rho_i \le b$, it follows that the total cost is bounded by $\tilde{O}(nb + n^2\tau_p + n^3 + n^2\log(\prod_{i=1}^k M(\tilde{z}_i)))$. The last summand is smaller than $n^2 \cdot 8\lambda$ according to step 6, and $\lambda < 2\|p\|/|p_n| < 2(n+1)2^{\tau_p}$. This shows the claim. □

We now prove correctness of the certification algorithm. In particular, we show that Inequality (13) implies that the disk Δ_i contains the same number of roots of the polynomials $\hat{p}$ and p.

[9] In fact, we compute an interval I_j of size less than 2^{-K} such that $|z_i^* - \hat{z}_j| \in I_j$, and then consider the product $\prod_j I_j$.

Lemma 9. (1) For all points $x \in \operatorname{bd}\Delta_i$, it holds that

$$|\hat{p}(x)| \geq \frac{1}{8}|\hat{p}(z_i^*)| .$$

(2) If Inequality (13) holds for all i, then Δ_i isolates a root of z_i of p of multiplicity $m_i = \operatorname{mult}(z_i, p) = |C_i|$.

(3) If $b \geq b_0$, then

$$\frac{|\hat{p}(z_i^*)|}{|p_n|} > \left(\frac{\min(256,\sigma_i)}{1024n}\right)^{m_i} \cdot \frac{|P_i|}{8} \geq 32 \cdot 2^{-b_0}\lambda M(\tilde{z}_i)^n$$

Proof. For a fixed $\hat{z}$, let x be the farthest point on $\operatorname{bd}\Delta_i$ from $\hat{z}$, and let y be the nearest. For $i \neq j$, we have

$$|x-\hat{z}| \leq |x-\tilde{z}_i| + |\tilde{z}_i - \tilde{z}_j| + |\tilde{z}_j - \hat{z}| \leq (1+1/n)|\tilde{z}_i - \tilde{z}_j| \text{ , and}$$
$$|y-\hat{z}| \geq |\tilde{z}_i - \tilde{z}_j| - |y-\tilde{z}_i| - |\tilde{z}_j - \hat{z}| \geq (1-1/n)|\tilde{z}_i - \tilde{z}_j| \text{ .}$$

Similarly, for $i = j$:

$$|x-\hat{z}| \leq |x-\tilde{z}_i| + |\tilde{z}_i - \hat{z}| \leq (1+1/n)nr_i \text{ , and}$$
$$|y-\hat{z}| \geq |y-\tilde{z}_i| - |\tilde{z}_i - \hat{z}| \geq (1-1/n)nr_i \text{ .}$$

Consequently, for any $x, y \in \operatorname{bd}\Delta_i$, it holds that

$$|\hat{p}(x)| = |p_n| \cdot \prod_{\ell=1}^{n} |x - \hat{z}_\ell| \geq \left(\frac{1-1/n}{1+1/n}\right)^n |p_n| \cdot \prod_{\ell=1}^{n} |y-\hat{z}_\ell| \geq \frac{1}{8}|\hat{p}(y)| \text{ .}$$

This shows the first claim.

We turn to the second claim. Since $nr_i < 1/n$, we have $|x| < (1+1/n)M(\tilde{z}_i)$ for all $x \in \operatorname{bd}\Delta_i$. Now, if $|\hat{p}(z_i^*)/p_n| > 64 \cdot 2^{-b}\lambda M(\tilde{z}_i)^n$, then

$$|\hat{p}(x)| > \frac{|\hat{p}(z_i^*)|}{8} > 2|p_n|\lambda 2^{-b} M(x)^n > \|p\| 2^{-b} M(x)^n \geq \|\hat{p} - p\| M(x)^n \geq |\hat{p}(x) - p(x)|.$$

Hence, according to Rouché's theorem. Δ_i contains the same number (namely, $|C_i|$) of roots of p and $\hat{p}$. If this holds for all disks Δ_i, then each of the disks must contain exactly one root since p has k distinct roots. In addition, the multiplicity of each root equals the number $|C_i|$ of approximations within Δ_i.

It remains to show the third claim. Since $b \geq b_0$, it follows that $\min(1/(2n^2), \sigma_i/(512n^2)) \leq r_i \leq \min(1/n^2, \sigma_i/(64n^2))$ and $|\tilde{z}_i - z_i| < r_i$; cf. the remark following the definition of r_i in (12). Thus,

$$\begin{aligned}
|\hat{p}(z_i^*)| &\geq |p(z_i^*)| - 2^{-b}\,\|p\| \cdot M(z_i^*)^n \\
&= |p(z_i + (\tilde{z}_i - z_i + nr_i))| - 2^{-b}\,\|p\| \cdot M(z_i^*)^n \\
&\geq ((n-1)r_i)^{m_i}|p_nP_i|/4 - 4 \cdot 2^{-b}\,\|p\| M(z_i)^n \\
&\geq \left(\frac{(n-1)\min(256,\sigma_i)}{512n^2}\right)^{m_i} \cdot \frac{|p_nP_i|}{4} - 4 \cdot 2^{-b}\,\|p\| M(z_i)^n \\
&\geq \left(\frac{\min(256,\sigma_i)}{1024n}\right)^{m_i} \cdot \frac{|p_nP_i|}{4} - 4 \cdot 2^{-b}\,\|p\| M(z_i)^n,
\end{aligned}$$

where the first inequality is due to $|(p-\hat{p})(x)| < 2^{-b}\,\|p\| \cdot M(x)^n$, the second inequality follows from $|\tilde{z}_i - z_i + nr_i| \leq (n+1)r_i \leq \sigma_i/n$, Lemma 2 and $M(z_i^*) < (1+1/n) \cdot M(z_i)$, and the third inequality follows from $r_i \geq \min(\frac{1}{2n^2}, \frac{\sigma_i}{512n^2})$. In addition, we have

$$2^{-b}\|p\|M(z_i)^n \leq \left(\frac{\min(256,\sigma_i)}{1024n}\right)^{m_i} \cdot \frac{|p_nP_i|}{4096}, \tag{14}$$

since

$$\begin{aligned}2^{-b}\|p\|M(z_i)^n &\leq 2^{-b/8}\cdot 2^{-b/2}\cdot 2^{\tau_p}|p_n|\cdot(n+1)\cdot M(z_i)^n\\ &\leq \frac{|P_i|}{(n+1)2^{2n\Gamma_p+8n}}\left(\frac{\min(256,\sigma_i)}{1024n}\right)^{m_i} 2^{\tau_p}|p_n|(n+1)M(z_i)^n\\ &\leq \left(\frac{\min(256,\sigma_i)}{1024n}\right)^{m_i}\cdot\frac{|p_nP_i|}{2^{7n-1}} \leq \left(\frac{\min(256,\sigma_i)}{1024n}\right)^{m_i}\cdot\frac{|p_nP_i|}{4096},\end{aligned}$$

where the second inequality follows from (11), (10), and (7) [10], and the third inequality follows from $\tau_p \leq n\Gamma_p + n + 1$ (Lemma 1) and $M(z_i)^n \leq 2^{n\Gamma_p}$. Finally,

$$\frac{|\hat{p}(z_i^*)|}{|p_n|} > \left(\frac{\min(256,\sigma_i)}{1024n}\right)^{m_i}\cdot\frac{|P_i|}{8} \geq 512\cdot 2^{-b}\frac{\|p\|}{|p_n|}M(z_i)^n \geq 64\cdot 2^{-b}\lambda M(\tilde{z}_i)^n,$$

where the first and the second inequality follow from (14) and the third inequality holds since λ is a 2-approximation of $\|p\|/|p_n|$ and $|z_i|^n \leq (1+1/n)^n|\tilde{z}_i|^n \leq 4|\tilde{z}_i|^n$. Since the values σ_i, m_i, and P_i do not depend on the choice of b, and the above inequality holds for any $b \geq b_0$, it follows that $512\cdot 2^{-b}\frac{\|p\|}{|p_n|}M(z_i)^n \geq 64\cdot 2^{-b_0}\lambda M(\tilde{z}_i)^n$. □

Lemma 10. There exists a b^* upper bounded by

$$O\left(n\log n + n\Gamma_p + \sum_{i=1}^k\left(\log M(P_i^{-1}) + m_i\log M(\sigma_i^{-1})\right)\right)$$

such that the certification step succeeds for any $b > b^*$. The total cost in the certification algorithm (i.e. for all iterations until we eventually succeed) is bounded by

$$\tilde{O}\left(n^3 + n^2\tau_p + n\cdot\sum_{i=1}^k\left(\log M(P_i^{-1}) + m_i\log M(\sigma_i^{-1})\right)\right)$$

bit operations.

Proof. Let $b \geq b_0 + 1$. Then, due to Lemma 9,

$$|\hat{p}(z_i^*)/p_n| > \left(\frac{\min(256,\sigma_i)}{1024n}\right)^{m_i}\cdot\frac{|P_i|}{8} \geq 64\cdot 2^{-b_0}\lambda M(\tilde{z}_i)^n \geq 128\cdot 2^{-b}\lambda M(\tilde{z}_i)^n$$

Thus, in order to verify inequality (13), it suffices to evaluate $|\hat{p}(z_i^*)/p_n|$ to an error of less than $|\hat{p}(z_i^*)/2p_n|$. It follows that we succeed for some ρ_i with

$$\rho_i = O(m_i\log n + m_i\max(1,\log\sigma_i^{-1}) + \log\max(1,|P_i|^{-1})).$$

In Step 3 of the certification algorithm, we require that the sum over all ρ_i does not exceed b. Hence, we eventually succeed in verifying the inequality (13) for all i if b is larger than some b^* with

$$\begin{aligned}b^* &= O(b_0 + \sum_i m_i\log n + \sum_i(\log M(P_i^{-1}) + m_i\log M(\sigma_i^{-1})))\\ &= O(n\log n + n\Gamma_p + \sum_i(\log M(P_i^{-1}) + m_i\log M(\sigma_i^{-1}))).\end{aligned}$$

[10] Observe $2^{-b/(2m_i)} \leq \min(\frac{1}{2n^2}, \frac{\sigma_i}{1024n^2}) \leq \frac{\min(256,\sigma_i)}{1024n}$.

For the bound for the overall cost, we remark that, for each b, the certification algorithm needs $\tilde{O}(n^3+nb+n^2\tau_p)$ bit operations due to Lemma 8. Thus, the above bound follows from the fact that that we double b in each step and that the certification algorithm succeeds under guarantee for all $b>b^*$. $\square$

2.3. *Complexity of Root Isolation*

We now turn to the complexity analysis of the root isolation algorithm. In the first step, we provide a bound for general polynomials p with real coefficients. In the second step, we give a simplified bound for the special case, where p has integer coefficients. We also give bounds for the number of bit operations that is needed to refine the isolating disks to a size less than $2^{-\kappa}$, with κ an arbitrary positive integer.

Theorem 3. Let $p(x)\in\mathbb{C}[x]$ be a polynomial as defined in Section 2.1. We assume that the number k of distinct roots of p is given. Then, for all $i=1,\dots,k$, the algorithm from Section 2.2 returns an isolating disk $\Delta(\tilde{z}_i,R_i)$ for the root z_i together with the corresponding multiplicity m_i, and $R_i<\frac{\sigma_i}{64n}$.

For that, it uses a number of bit operations bounded by

$$\tilde{O}\left(n^3+n^2\tau_p+n\cdot\sum_{i=1}^{k}\left(\log M(P_i^{-1})+m_i\log M(\sigma_i^{-1})\right)\right) \tag{15}$$

The algorithm needs an absolute L-approximation of p, with L bounded by

$$\tilde{O}\left(n\Gamma_p+\sum_{i=1}^{k}\left(\log M(P_i^{-1})+m_i\log M(\sigma_i^{-1})\right)\right). \tag{16}$$

Proof. For a fixed b, let us consider the cost for each of the steps in the algorithm:

- Steps 1-3, 5 and 6 do not use more than $\tilde{O}(n^2\Gamma_p+nb)$ bit operations,
- Step 4 and 7 do not use more than $\tilde{O}(n^2\Gamma_p+nb)$ bit operations (Corollary 1 and Lemma 7), and
- Step 8 and 9 use a number of bit operations bounded by (15) (Lemma 10).

In addition, for a fixed b, the oracle must provide an absolute L-approximation of p, with $L=\tilde{O}(n\Gamma_p+b)$, in order to compute the bound Γ for Γ_p, to compute the 2-approximation λ of $\|p\|/|p_n|$, and to run Pan's algorithm. The algorithm succeeds in computing isolating disks if $b>b^*$ with a b^* as in Lemma 10. Since we double b in each step, we need at most $\lceil\log b^*\rceil$ iterations and the total cost for each iteration is bounded by (15). This shows the complexity result.

It remains to prove the bound for R_i. When the clustering succeeds, it returns disks $D_i=\Delta(\tilde{z}_i,r_i)$ with $\min(\frac{1}{2n^2},\frac{\tilde{\sigma}_i}{256n^2})\le r_i\le\min(\frac{1}{n^2},\frac{\tilde{\sigma}_i}{128n^2})$ for all $i=1,\dots,m$. It follows that $R_i=n\cdot r_i\le\frac{\tilde{\sigma}_i}{128n}$, and thus $|z_i-z_j|\ge|\tilde{z}_i-\tilde{z}_j|-|z_i-\tilde{z}_i|-|z_j-\tilde{z}_j|>|\tilde{z}_i-\tilde{z}_j|\cdot(1-1/(64n))>|\tilde{z}_i-\tilde{z}_j|/2$ for all i,j with $i\neq j$. We conclude that $\sigma_i>\tilde{\sigma}_i/2\ge 64nR_i$. $\square$

We remark that the bound (15) can also be reformulated in terms of values that exclusively depend on the degree n and the geometry of the roots (i.e. their absolute values and their distances to each other). Namely, according to Lemma 1, we have $\tau_p\le n+1+\log\frac{\mathrm{Mea}(p)}{|p_n|}$, and the latter expression only involves the degree and the absolute values of the roots of p. This yields the bound (2) from the introduction.

In the next step, we show that combining our algorithm with Pan's factorization algorithm also yields a very efficient method to further refine the isolating disks.

Theorem 4. Let $p(x)$ be a polynomial as in Theorem 3, and κ be a given positive integer. We can compute isolating disks $\Delta_i(\tilde{z}_i, R_i)$ with radius $R_i < 2^{-\kappa}$ in a number of bit operations bounded by

$$\tilde{O}\left(n^3 + n^2\tau_p + n \cdot \sum_{i=1}^{k} \left(\log M(P_i^{-1}) + m_i \log M(\sigma_i^{-1})\right) + n\kappa \cdot \max_{1\le i\le k} m_i\right). \tag{17}$$

For that, we need an absolute L-approximation of p with L bounded by

$$\tilde{O}\left(n\Gamma_p + \sum_{i=1}^{k} \left(\log M(P_i^{-1}) + m_i \log M(\sigma_i^{-1})\right) + n\kappa \cdot \max_{1\le i\le k} m_i\right).$$

Proof. As a first step, we use the algorithm from Section 2.2 to compute isolating disks $\Delta_i = \Delta(\tilde{z}_i, R_i)$ with $R_i \le \sigma_i/(64n)$. Each disk Δ_i contains the root z_i, $m_i = \operatorname{mult}(z_i, p)$ approximations $\hat{z} \in \{\hat{z}_1, \ldots, \hat{z}_n\}$ of z_i, and it holds that $\sigma_i/2 < \tilde{\sigma}_i < 2\sigma_i$. Let

$$\hat{P}_i := \prod_{j:\hat{z}_j \notin \Delta_i} (\tilde{z}_i - \hat{z}_j).$$

We claim that $1/2|P_i| < |\hat{P}_i| < 2|P_i|$. Since $|\tilde{z}_i - z_i| < \sigma_i/(64n)$ for all i, it holds that $(1 - \frac{1}{64n})|z_i - z_j| \le |\tilde{z}_i - \hat{z}| \le (1 + \frac{1}{64n})|z_i - z_j|$ for all $j \neq i$ and $\hat{z} \in \Delta_j$. Thus, $|\hat{P}_i|$ is a 2-approximation of $|P_i|$. Similar as in the certification step, we now use approximate interval arithmetic to compute a 2-approximation μ_i of $|\hat{P}_i|$, and thus a 4-approximation of $|P_i|$. A completely similar argument as in the proofs of Lemma 8 and Lemma 10 then shows that we can compute such μ_i's with less than $\tilde{O}(n^3 + n^2\tau_p + n\sum_i \log M(P_i^{-1}))$ bit operations. Now, from the 2- and 4-approximations of σ_i and $|P_i|$, we can determine a b_κ such that

- the properties (6) to (9) are fulfilled, and
- $2^{-b/(2m_i)} < 2^{-\kappa}$.

Then, from Corollary 1 and Lemma 3, we conclude that Pan's factorization algorithm (if run with $b \ge b_\kappa$) returns, for all i, m_i approximations $\hat{z}$ of z_i with $|\hat{z} - z_i| < 2^{-b/(2m_i)} < 2^{-\kappa}$. Thus, for each i, we can simply choose an arbitrary approximation $\hat{z} \in \Delta_i$ and return the disk $\Delta(\hat{z}, 2^{-\kappa})$ which isolates z_i. The total cost splits into the cost for the initial root isolation and the cost for running Pan's Algorithm with $b = b_\kappa$. Since the latter cost is bounded by $\tilde{O}(nb_\kappa + n^2\Gamma_p)$, the bound (17) follows. □

Finally, we apply the above results to the important special case, where we aim to isolate the roots of a polynomial with integer coefficients.

Theorem 5. Let $p(x) \in \mathbb{Z}[x]$ be a polynomial of degree n with integer coefficients of size less than 2^τ. Then, we can compute isolating disks $\Delta(\tilde{z}_i, R_i)$, with $R_i < \frac{\sigma_i}{64n}$, for all roots z_i together with the corresponding multiplicities m_i using

$$\tilde{O}(n^3 + n^2\tau) \tag{18}$$

bit operations. For a given positive integer κ, we can further refine the disks Δ_i to a size of less than $2^{-\kappa}$ with a number of bit operations bounded by

$$\tilde{O}(n^3 + n^2\tau + n\kappa). \tag{19}$$

Proof. In a first step, we compute the square-free part $p^* = p/\gcd(p, p')$ of p. According to [**?** , §11.2], we need $\tilde{O}(n^2\tau)$ bit operations for this step, and p^* has integer coefficients of bitsize $O(n + \tau)$. The degree of p^* yields the number k of distinct roots of p. In order to use our root isolation algorithm from Section 2.2, we divide p by its leading coefficients p_n to meet the requirement that the leading coefficient has absolute value in $[1/4,1]$. Obviously, the roots are not affected by this normalization step.

Now, in order to derive the bound in (18), we have to reformulate the bound from (15) in terms of the degree n and the bitsize τ of p. We first use [15, Theorem 2] to show that $\sum_{i=1}^{k} m_i \log\max(1,\sigma_i^{-1}) = \tilde{O}(n^2+n\tau)$. Furthermore, we have $\tau_p \leq \tau$. Hence, it remains to show that $n\cdot\sum_{i=1}^{k}\log M(P_i^{-1}) = \tilde{O}(n^3+n^2\tau)$. For that, we consider a square-free factorization $p(x) = \prod_{l=1}^{n}(Q_l(x))^l$ with square-free polynomials $Q_l \in \mathbb{Z}[x]$ such that Q_l and p/Q_l^l are coprime for all $l=1,\dots,n$. Note that the roots of Q_l are exactly the roots of p with multiplicity l, and that Q_l is a constant for most l. We further denote $\bar{p} := p/\operatorname{lcf}(p)$ and $\bar{Q}_l := Q_l/\operatorname{lcf}(Q_l)$. Let S_l denote the set of roots of Q_l. Then, from the definition of P_i,

$$\begin{aligned}
\prod_{i:z_i\in S_l}|P_i| &= \prod_{i:z_i\in S_l}\prod_{j\neq i}|z_i-z_j|^{m_j}\\
&= \prod_{i:z_i\in S_l}\prod_{j\neq i:z_j\notin S_l}|z_i-z_j|^{m_j}\cdot\prod_{i:z_i\in S_l}\prod_{j\neq i:z_j\in S_l}|z_i-z_j|^{l}\\
&= \prod_{i\in S_l}|(\bar{p}/\bar{Q}_l^l)(z_i)|\cdot\prod_{i:z_i\in S_l}|(\bar{Q}_l)'(z_i)|^l\\
&= |\operatorname{res}(\bar{p}/\bar{Q}_l^l,\bar{Q}_l)|\cdot|\operatorname{res}(\bar{Q}_l,(\bar{Q}_l)')|^l\\
&= \frac{|\operatorname{res}(p/Q_l^l,Q_l)|}{|\operatorname{lcf}(Q_l)^{n-l\cdot\deg Q_l}(\operatorname{lcf}(p/Q_l^l))^{\deg Q_l}|}\cdot\left|\frac{\operatorname{res}(Q_l,Q_l')}{\operatorname{lcf}(Q_l)^{2\deg Q_l-1}(\deg Q_l)^{\deg Q_l}}\right|^l\\
&\geq \frac{1}{|\operatorname{lcf}(Q_l)|^{n-l}\cdot|\operatorname{lcf}(p)|^{\deg Q_l}\cdot n^{l\deg Q_l}}
\end{aligned}$$

where $\operatorname{res}(f,g)$ denotes the resultant [11] of two polynomials f and g. For the last inequality, we used that $\operatorname{res}(p/Q_l^l,Q_l)\in\mathbb{Z}$ and $\operatorname{res}(Q_l,Q_l')\in\mathbb{Z}$. Taking the product over all l yields

$$\prod_{i=1}^{k}|P_i| \geq \left|\frac{1}{\prod_{l=1}^{n}(\operatorname{lcf}(Q_l)^{n-l}\cdot\operatorname{lcf}(p)^{\deg Q_l}\cdot n^{l\deg Q_l})}\right| \geq \frac{1}{|\operatorname{lcf}(p)|^{2n}\cdot n^n} \geq 2^{-2n\tau-n\log n}.$$

Note that, for any i, we also have

$$|P_i| = \frac{|p^{(m_i)}(z_i)|}{m_i!p_n} < \frac{m_i!2^{\tau}(n+1)M(z_i)^n}{m_i!|p_n|} \leq n2^{\tau+1}M(z_i)^n,$$

and, thus,

$$\sum_{i=1}^{k}\log M(P_i^{-1}) = \tilde{O}(n\tau + n\sum_{i=1}^{k}\log M(z_i)) = \tilde{O}(n\tau),$$

where we used that $\sum_i \log M(z_i)) \leq \log\operatorname{Mea}(p) \leq \log\|p\| < \log(n+1)+\tau$. This shows (18).

For the bound in (19) for the cost of refining the isolating disks $\Delta_i(\tilde{z}_i,R_i)$ to a size of less than $2^{-\kappa}$, we consider the square-free part p^*. Note that the disks Δ_i obtained in the first step are obviously also isolating for the roots of p^* (p and p^* have exactly the same distinct roots) and that $R_i < \sigma(z_i,p)/(64n) = \sigma(z_i,p^*)/(64n) \leq \sigma(z_i,p^*)/(64\deg p^*)$. Thus, proceeding in completely analogous manner as in the proof of Theorem 4 (with the square-free part p^* instead of

[11] For univariate polynomials

$$\operatorname{res}(f,g) = \operatorname{lcf}(f)^{\deg g}\operatorname{lcf}(g)^{\deg f}\prod_{(x,y):f(x)=g(y)=0}(x-y) = \operatorname{lcf}(f)^{\deg g}\prod_{x:f(x)=0}g(x).$$

p) shows that we need $\tilde{O}(n^3 + n^2\tau + n\kappa)$ bit operations for the refinement. This proves the second claim. $\square$

2.4. Well-separated Clusters of Roots

We now turn to the problem of computing well-separated clusters of roots. We no longer insist that the clusters are in one-to-one correspondence with the roots, but may have clusters containing more than one root as long as the clusters are well-separated. Well-separated means that the diameter of each cluster is much smaller than the distance from the cluster to the nearest distinct cluster. We also need to impose an upper bound on the diameter of any cluster to make the problem non-trivial. Otherwise, it would be allowed to return a single cluster, e.g., the disk centered at the origin and having radius $4\max_i |p_i| / |p_n|$, containing all roots. Recall that this disk contains all roots of p (Lemma 1).

Renegar's algorithm [39] computes clusters of radius ε, where $\varepsilon > 0$ is an input parameter. More precisely, it computes $\tilde{z}_1$ to $\tilde{z}_j$ (the number of clusters is not predetermined) and multiplicities m_i such that $\sum_i m_i = n$, the disks $\Delta_i = \Delta(\tilde{z}_i, \varepsilon)$ are disjoint, and Δ_i contains exactly m_i roots of p. He uses subdivision and Newton iteration for root approximation, the Shur-Cohn method [24, Theorem 6.8b] for determining whether a disk contains a root, and an approximate winding number algorithm for estimating the number of zeros in a disk. The arithmetic complexity (= number of arithmetic operations) is analyzed and shown to be nearly optimal. The author also states that "his algorithm will not fare well in the bit-complexity model".

Yakoubsohn and Giusti et. al. [19, 52] show how to approximate a single cluster of zeros. Given a good starting point, they derive an estimate for the number of zeros in the cluster from the convergence rate of Newton's method. They verify the number of roots in a cluster by an inclusion test based on Rouché's theorem. Schröder's variant of Newton's method is used to improve the approximation of the cluster. In the case of a multiple root of known multiplicity it is known to converge quadratically [24]. They show that, in the case of a cluster of roots, it is still quadratic provided the iteration is stopped sufficiently early. They propose a method for stopping the iteration at a distance from the cluster which is on the order of its diameter.

We modify our algorithm as follows. The input to the algorithm is the polynomial p. In the clustering algorithm (Section 2.2.2), we drop step (6), i.e., we allow the algorithm to generate any number of clusters. After the clustering, we proceed to the verification step. If the verification step succeeds (this includes a check that the disks have radius at most $4\max_i |p_i| / |p_n|$), we output the clusters determined in the clustering step. Otherwise, we double b and repeat. The modified algorithm has the following properties:

(1) If it returns disks D_1 to D_j and associated multiplicities m_1 to m_j, then $\sum_i m_j = n$, Δ_j contains m_j root approximations and m_j roots of p counted with multiplicity, and the disks with the n-fold radii are pairwise disjoint.
(2) The algorithms stops at the latest when the precision exceeds b_0, where b_0 is as in the preceding section.
(3) The bit complexity of the algorithm is as stated in (15).

3. Curve Analysis

In this section, we show how to integrate our approach to isolate and approximate the roots of a univariate polynomial in an algorithm to compute a cylindrical algebraic decomposition [2, 3, 4, 9, 12, 14, 21, 29, 49]. More specifically, we apply the results from the previous section to a recent algorithm, denoted TOPNT, from [4] for computing the topology of a real planar algebraic curve. This yield a bound on the expected number of bit operations for computing the topology of a real planar algebraic curve that improves the currently best bound [29] from $\tilde{O}(n^9\tau+n^8\tau^2)$ (deterministic) to $\tilde{O}(n^6+n^5\tau)$ (randomized). Isolating the real-valued solutions of a bivariate polynomial system $g(x,y)=h(x,y)=0$ can be reduced to the problem of computing the topology of the algebraic curve $f:=g^2+h^2$ whose degree is comparable to the degree of the polynomials g and h. Based on the latter observation, we derive a bound on the expected number of bit operations for solving a bivariate polynomial system that improves the best known bound [15] from $\tilde{O}(n^8+n^7\tau)$ (deterministic) to $\tilde{O}(n^6+n^5\tau)$; see Theorem 7.

We also remark that an implementation of algorithm TOPNT is available [4]. The implementation uses a variant of the Aberth-Ehrlich method for root isolation [31, 42] and shows great efficiency in practice.

3.1. Review of the Algorithm TOPNT

For the sake of a self-contained representation, we briefly review the algorithm TOPNT. For more details and the corresponding proofs, we refer to [4]. The input of the algorithm is a bivariate polynomial $f\in\mathbb{Z}[x,y]$ of total degree n with integer coefficients of magnitude 2^τ or less. The polynomial defines an algebraic curve

$$C:=\{(x,y)\in\mathbb{C}^2 : f(x,y)=0\}\subseteq\mathbb{C}^2.$$

The algorithm returns a planar straight-line graph $\mathcal{G}$ embedded in $\mathbb{R}^2$ that is *isotopic*[12] *to the real part* $C_{\mathbb{R}}:=C\cap\mathbb{R}^2$ of C.

In the first step (the **shearing step**), we choose an $s\in\mathbb{Z}$ at random (initially, consider $s=0$) and consider the sheared curve

$$C_s:=\{(x,y)\in\mathbb{C}^2 : f_s(x,y):=f(x+s\cdot y,y)=0\}.$$

Then, any planar graph isotopic to the real part $C_{s,\mathbb{R}}:=C_s\cap\mathbb{R}$ of C_s is also isotopic to $C_{\mathbb{R}}$, and vice versa. We choose s such that the leading coefficient $\operatorname{lcf}(f_s(x,y);y)$ (with respect to y) of the defining polynomial $f_s(x,y)$ of C_s is a constant. This guarantees that $C_{s,\mathbb{R}}$ has no vertical asymptote and that it contains no vertical line. By abuse of notation, we write $C=C_s$ and $f=f_s$ throughout the following considerations.

In the **projection step**, the x-critical points of C (i.e. all points $(x_0,y_0)\in C$ with $f_y(x_0,y_0)=0$, where $f_y:=\frac{\partial f}{\partial y}$) are projected onto the real x-axis by means of a resultant computation. More precisely, we compute

- $R:=\operatorname{res}(f,f_y;y)\in\mathbb{Z}[x]$,
- its square-free part $R^*:=R/\gcd(R,R')$,
- isolating intervals $I_1,\dots,I_m$ for the real roots $\alpha_1,\dots,\alpha_m$ of R^*,
- the multiplicity $m_i:=\operatorname{mult}(\alpha_i,R)$ of α_i as a root of R for all $i=1,\dots,m$, and
- arbitrary separating values $\beta_0,\dots,\beta_{m+1}\in\mathbb{R}$ with $\alpha_m<\beta_{m+1}$, and $\beta_{i-1}<\alpha_i<\beta_i$ for all $i=1,\dots,m$.

We further compute

[12] We actually consider the stronger notion of an *ambient isotopy*, but omit the "ambient". $\mathcal{G}$ is ambient isotopic to $C_{\mathbb{R}}$ if there is a continuous mapping $\phi:[0,1]\times\mathbb{R}^2\mapsto\mathbb{R}^2$ with $\phi(0,\cdot)=\mathrm{id}_{\mathbb{R}^2}$, $\phi(1,C_{\mathbb{R}})=\mathcal{G}$, and $\phi(t_0,\cdot)$ is a homeomorphism for each $t_0\in[0,1]$.

- $f_x^* := \frac{f_x}{\gcd(f_x,f_y)}$ and $f_y^* := \frac{f_y}{\gcd(f_x,f_y)}$,
- $Q := \operatorname{res}(f_x^*, f_y^*; y)$, and
- the multiplicity $l_i := \operatorname{mult}(\alpha_i, Q)$ of α_i as a root of Q for all $i = 1, \ldots, m$.

In the **lifting step**, we compute the fibers of C at the points α_i and β_i, that is, we isolate the roots of the polynomials $f_{\alpha_i}(y) := f(\alpha_i, y) \in \mathbb{R}[y]$ and $f_{\beta_i}(y) := f(\beta_i, y) \in \mathbb{R}[y]$. For that, we first compute the number of distinct complex roots of each of these polynomials, and then use the root isolator from Section 2. [13] Obviously, each polynomial $f_{\beta_i}(y)$ has $k(\beta_i) = \deg f_{\beta_i} = n$ distinct complex roots. The difficult part is to determine the number $k(\alpha)$ of distinct roots of $f_\alpha(y)$ for a root α of R^*. According to [4, (3.6)] and [4, Theorem 5],

$$k^+(\alpha) := n - \operatorname{mult}(\alpha, R) + \operatorname{mult}(\alpha, Q) \geq k(\alpha), \tag{20}$$

and, for a generic shearing factor s (more precisely, for all but $n^{O(1)}$ many s), the equality $k^+(\alpha) = k(\alpha)$ holds for all roots α of R^*. Summation over all complex roots of R^* then yields

$$K^+ := \sum_{\alpha:R^*(\alpha)=0} k^+(\alpha) = n \cdot \deg R^* - \deg R + \deg \gcd(R^\infty, Q) \geq \sum_{\alpha:R^*(\alpha)=0} k(\alpha) =: K, \text{ and}$$
$$K = K^+ \text{ for generic } s,$$

where $\gcd(R^\infty, Q)$ is defined as the product of all common factors of R and Q with multiplicities according to their occurrence in Q. The crucial idea is now to compare the upper bound K^+ with a lower bound K^- which also equals K up to a non-generic choice of some parameters. In order to understand the computation of K^-, we first consider the exact computation of K: Let $\operatorname{Sres}_i(f, f_y; y) \in \mathbb{Z}[x,y]$ denote the *i-th subresultant polynomial* of f and f_y (with respect to y), and $\operatorname{sr}_i(x) := \operatorname{sres}_i(f, f_y; y) \in \mathbb{Z}[x]$ its leading coefficient. In particular, we have $R = \operatorname{sres}_0(f, g; y) = \operatorname{res}(f, f_y; y)$. We define:

$$\begin{aligned} S_0 &:= R^*, & S_i &:= \gcd(S_{i-1}, \operatorname{sr}_i) \\ R_1 &:= \frac{S_0}{S_1} = \frac{S_0}{\gcd(S_0, S_1)}, & R_i &:= \frac{S_{i-1}}{S_i} = \frac{\gcd(S_0, \ldots, S_{i-1})}{\gcd(S_0, \ldots, S_{i-1}, S_i)}, \end{aligned} \tag{21}$$

where $i = 1, \ldots, n$. Then, $\prod_{i\geq 1} R_i$ constitutes a factorization of R^* such that $R_i(\alpha) = 0$ if and only if $f(\alpha, y)$ has exactly $n - i$ distinct complex roots; see [4, Section 3.2.2] for details. Hence, we have $K = \sum_{i\geq 1}(n-i) \cdot \deg R_i$. We do not carry out the latter computation of K over the integer domain, but over a modular prime field which yields a lower bound K^- for K. More precisely, we choose a prime p at random, compute the modular images $\operatorname{sr}_i^{(p)}(x) = \operatorname{sres}_i(f \bmod p, f_y \bmod p; y) \in \mathbb{Z}_p[x]$ of $\operatorname{sr}_i(x) \in \mathbb{Z}[x]$, and perform all computations from (21) in $\mathbb{Z}_p[x]$. This yields polynomials $R_i^{(p)} \in \mathbb{Z}_p[x]$. Now, [4, Lemma 4] shows that

$$K^- := \sum_{i\geq 1} (n-i) \cdot \deg R_i^{(p)} \leq K, \tag{22}$$

and $K^- = K$ for all but finitely many bad primes. [14] Hence, if $K^- < K^+$, we have either chosen a bad prime or a bad shearing factor. In this case, we start over with a new s and choose a new

[13] More precisely, we first compute some 2^t, with $\operatorname{lcf}(f_s(x,y); y) \leq 2^t \leq 4 \cdot \operatorname{lcf}(f_s(x,y); y)$, and apply the root isolator from Section 2 to the polynomial $2^{-t} \cdot f_{\alpha_i}(y)$ (and $2^{-t} \cdot f_{\beta_i}(y)$, respectively) which has leading coefficient of absolute value between $1/4$ and 1.

[14] In the computation of the S_i's and R_i's (over $\mathbb{Z}$), all intermediate results have integer coefficients of bitsize bounded by $(n\tau)^{O(1)}$. Since the product of N distinct primes is larger than $N! = 2^{\Omega(N \log N)}$, there exist at most $(n\tau)^{O(1)}$ many bad primes for which $K^- \neq K$.

prime p in the lifting step. If $K^- = K^+$, we know for sure that $K^+ = K$, and thus $k^+(\alpha) = k(\alpha)$ for all roots α of the resultant polynomial R.

We can now use our method from Section 2 to isolate all complex roots of the fiber polynomials $f_{\alpha_i}(y)$ and $f_{\beta_i}(y)$. Namely, we can ask for arbitrary good approximations of α_i and β_i (by refining corresponding isolating intervals), and thus for arbitrary good approximations of the coefficients of the fiber polynomials. In addition, we know the exact number of distinct roots of either polynomial. From the isolating regions in $\mathbb{C}$, we then derive isolating intervals for the real roots together with corresponding multiplicities. If one of the polynomials $f_{\alpha_i}(y)$ has more than one multiple real root, we start over and choose a new shearing factor s. Otherwise, we proceed with the final step.

Connection step. We remark that, except for finitely many s, each f_{α_i} has exactly one multiple root. The previous two steps already yield the vertices of the graph $\mathscr{G}$. Namely, these are exactly the points [15]

$$V(\mathscr{G}) := \{(x,y) \in \mathbb{R}^2 : \exists i \text{ with } x = \alpha_i \text{ or } x = \beta_i, \text{ and } f(x,y) = 0\}$$

Since each polynomial $f_\alpha(y)$ has exactly one multiple root, there exists a unique vertex v along each vertical line, where either the number of edges connecting v to the left or to the right may differ from one. Hence, connecting all vertices in an appropriate manner is straightforward; see [4, Section 3.2.3] for more details.

Remark. We remark that we use randomization at exactly two stages of the algorithm, that is, the choice of a shearing value s in the projection step and the choice of a prime p for computing the lower bound K^- for K in the lifting step. Let P denote the set of all prime numbers, then there exists a set $B \subset \mathbb{Z} \times P$ of "bad" pairs $(s,p) \in B$ for which success of the algorithm is not guaranteed, whereas, the algorithm returns the correct topology of C for all other pairs. There are at most $n^{O(1)}$ "bad" values for s that yield a non-generic position of the curve, and, for each of the remaining values for s, there exist at most $(n\tau)^{O(1)}$ many "bad" choices for p. Since we can generate a random prime of bit length L or less for the cost of $L^{O(1)}$ bit operations, [16] it follows that using $(\log(n\tau))^{O(1)}$ bit operations, we can pick a pair (s,n) such that, with probability $1/2$, the algorithm succeeds.

3.2. *Complexity Analysis*

Throughout the following considerations, we say that a polynomial $G \in \mathbb{Z}[x_1,\dots,x_k]$ with integer coefficients has *magnitude* (N,μ) if the total degree of G is upper bounded by N and all coefficients have absolute value 2^μ or less. In addition, we fix the following notations: For an arbitrary $\alpha \in \mathbb{C}$,

- we define $f_\alpha(y) := \sum_{i=0}^{n} f_{\alpha,i} y^i := f(\alpha,y) \in \mathbb{C}[y]$, where $f \in \mathbb{Z}[x,y]$ is our input polynomial. We further define $\tau_\alpha := \log \max_i |f_{\alpha,i}| \geq 0$ (notice that, for the considered shearing factors s, the leading coefficient $f_{\alpha,n}$ is a constant integer for all α).

[15] For a graph with rational vertices, you may replace each $x_0 = \alpha_i$ (or $x_0 = \beta_i$) by an arbitrary rational value in its corresponding isolating interval, and the same for each real root of $f_{x_0}(y)$.

[16] In order to generate a random prime of size less than 2^L, pick an integer of magnitude less than 2^L at random and test this integer for being prime. Since the cost for the latter test is polynomial [1] in L and since there exist [40] more than $2^L/(\ln(2)\cdot L+2)$ prime numbers of size less than 2^L for any $L \geq 6$, we can pick a random prime of bit length less than L with a number of bit operations that is polynomial in L.

- the number of distinct roots of f_α is denoted by $k(\alpha)$. We further denote $z_{\alpha,1},\dots,z_{\alpha,k(\alpha)}$ the distinct roots of f_α, and $m_{\alpha,i}$, with $i=1,\dots,k(\alpha)$, the corresponding multiplicities.
- $\sigma_{\alpha,i}$ denotes the separation of $z_{\alpha,i}$, and $P_{\alpha,i} := \prod_{j\neq i}(z_{\alpha,i}-z_{\alpha,j})^{m_{\alpha,j}}$.
- For an arbitrary polynomial $G\in\mathbb{C}[x]$, we denote $V(G)$ the set of all distinct complex roots of G, and $\mathscr{V}(G)$ the multiset of all complex roots (i.e. each root occurs a number of times according to its multiplicity).

We first prove the a couple of basic results which are needed for our analysis:

Lemma 11. For a fixed positive integer k, let $G\in\mathbb{Z}[x_1,\dots,x_k]$ be an integer polynomial of magnitude (N,μ). Then, each divisor $g\in\mathbb{Z}[x_1,\dots,x_k]$ of G has coefficients of bitsize $\tilde{O}(\mu+N)$.

Proof. We prove the claim via induction over k. For a univariate $G\in\mathbb{Z}[x_1]$, we remark that $\operatorname{Mea}(g)\le\operatorname{Mea}(G)\le\|G\|_2\le 2^{\mu+1}N$, and thus the absolute value of each coefficient of g is bounded by $2^N\operatorname{Mea}(g)\le 2^{N+\mu+1}N$.

For the general case, we write

$$g(x_1,\dots,x_k)=\sum_{\lambda=(\lambda_1,\dots,\lambda_{k-1})} a_\lambda(x_k)x_1^{\lambda_1}\cdots x_{k-1}^{\lambda_{k-1}},\ \text{with } a_\lambda\in\mathbb{Z}[x_k].$$

For a fixed $\bar{x}_k\in\{0,\dots,N\}$, the polynomial $g(x_1,\dots,x_{k-1},\bar{x}_k)\in\mathbb{Z}[x_1,\dots,x_{k-1}]$ is a divisor of $G(x_1,\dots,x_{k-1},\bar{x}_k)\in\mathbb{Z}[x_1,\dots,x_{k-1}]$. Since $|\bar{x}_k|^N\le N^N=2^{N\log N}$ and $a_\lambda(x_k)$ has degree N or less, it follows that $G(x_1,\dots,x_{k-1},\bar{x}_k)$ has bitsize $O(N\log N+\mu)$. Hence, from the induction hypothesis, we conclude that the polynomial $g(x_1,\dots,x_{k-1},\bar{x}_k)$ has coefficients of bitsize $\tilde{O}(\mu+N)$, and thus $a_\lambda(i)\in\mathbb{Z}$ has bitsize $\tilde{O}(\mu+N)$ for all $i=0,\dots,N$ and all λ. Since a_λ is a polynomial of degree at most N, it follows that a_λ is uniquely determined by the values $a_\lambda(i)$, and thus Lagrange interpolation yields

$$a_\lambda(x)=\sum_{i=0}^{N}a_\lambda(i)\cdot\frac{x\cdot(x-1)\cdots(x-i+1)(x-i-1)\cdots(x-N)}{i\cdot(i-1)\cdots1\cdot(-1)\cdots(i-N)}$$

Expanding the numerator of the fraction yields a polynomial with coefficients of absolute value $2^{O(N\log N)}$, and thus each coefficient of $a_\lambda(x_k)$ has bitsize $\tilde{O}(\mu+N)$ because $a_\lambda(i)$ has bitsize $\tilde{O}(\mu+N)$ and there are $N+1$ summands. This proves the claim. □

In addition, we need a bound on the bit complexity of computing the greatest common divisor of two univariate polynomials with integer coefficients. For a proof of the following result, we refer to [51, §11.2].

Lemma 12. Let $F,G\in\mathbb{Z}[x]$ be two univariate polynomials of magnitude (N,μ).
- Computing $H:=\gcd(F,G)$ uses $\tilde{O}(N^2\mu)$ bit operations.
- Given a polynomial $H\in\mathbb{Z}[x]$ that divides F, computing F/H uses $\tilde{O}(N\mu)$ bit operations.

In the projection step of TopNT, we also have to compute the greatest common divisor of two bivariate polynomials. Although it is not very difficult to derive a reasonable good bound on the bit complexity of the latter problem, it seems that no complexity results for *deterministic* algorithms are published so far. The following lemma provides such a result:

Lemma 13. Let $F,G\in\mathbb{Z}[x,y]$ be two bivariate polynomials of magnitude (N,μ). Then, we can compute $H(x,y):=\gcd(F,G)$ with $\tilde{O}(N^6+N^5\mu)$ bit operations.

Proof. Throughout the following considerations, we say that two polynomials p_1, p_2 in $\mathbb{Z}[x]$ (or in $\mathbb{Z}[x,y]$) are equivalent (written as $p_1 \simeq p_2$) if there exists an integer $\lambda \in \mathbb{Z}$ such that $p_1 = \lambda \cdot p_2$ or $p_2 = \lambda \cdot p_1$. Let $F(x,y) = f_l(x) \cdot y^l + \cdots + f_0(x)$ and $G(x,y) = g_k(x) \cdot y^k + \cdots + g_0(x)$, with $l, k \le N$ and polynomials $f_i, g_j \in \mathbb{Z}[x]$. For an arbitrary but fixed $s \in \mathbb{Z}$, it holds that

$$H(x,y) = \gcd(F,G) = \hat{H}(x - s \cdot y, y), \text{ where } \hat{H}(x,y) := \gcd(F(x + s \cdot y, y), G(x + s \cdot y, y)).$$

Namely, for each divisor $d(x,y) \in \mathbb{Z}[x,y]$ of F and G, $d(x \pm s \cdot y, y)$ is also a divisor of $F(x \pm s \cdot y, y)$ and $G(x \pm s \cdot y, y)$, respectively. Thus, for computing $H(x,y)$, it suffices to compute $\hat{H}$ for an arbitrary integer $s_0 \in \mathbb{Z}$ and to replace x by $x - s_0 \cdot y$. We first determine an integer s_0 such that both polynomials $\hat{F} := F(x + s_0 y, y)$ and $\hat{G} := G(x + s_0 y, y)$ have constant leading coefficients with respect to y. That is,

$$\hat{F}(x,y) = \hat{f}_m(x) \cdot y^m + \cdots + \hat{f}_0(x) \text{ and } \hat{G}(x,y) = \hat{g}_n(x) \cdot y^n + \cdots + \hat{g}_0(x), \qquad (23)$$
$$\text{with } \hat{f}_i, \hat{g}_j \in \mathbb{Z}[x] \text{ and } \hat{f}_m, \hat{g}_n \in \mathbb{Z}.$$

Considering s as an indeterminate variable, computing $F(x + s \cdot y, y) \in \mathbb{Z}[s,x,y]$ and $G(x + s \cdot y, y) \in \mathbb{Z}[s,x,y]$ uses $\tilde{O}(N^4 + N^3\mu)$ bit operations because computing $(x + sy)^i$ for all $i = 0, \ldots, N$, needs $\tilde{O}(N^3)$ bit operations, and computing $f_i(x) \cdot (x + sy)^i$ and $g_i(x) \cdot (x + sy)^i$ for a fixed i needs $\tilde{O}(N^2(N + \mu))$ bit operations. The leading coefficients of $F(x + sy, y)$ and $G(x + sy, y)$ with respect to y are univariate polynomials in s of magnitude $(N, O(N \log N + \mu))$. Thus, computing an integer s_0, with $|s_0| \le N$, such that both of the latter univariate polynomials do not vanish needs at most $\tilde{O}(N^2\mu + N^3)$ bit operations (polynomial evaluation at the $2N + 1$ points $s = -N, -N+1, \ldots, -1, 0, 1, \ldots, N$). It follows that computing an s_0, with $|s_0| \le N$, which fulfills the desired properties from (23) needs $\tilde{O}(N^4 + N^3\mu)$ bit operations. Throughout the following considerations, we can further assume that there exists no integer different from ± 1 which divides $\hat{F}$ and $\hat{G}$. Namely, with $\tilde{O}(N^2(N + \mu))$ bit operations, we can divide $\hat{F}$ and $\hat{G}$ by the greatest common divisor λ (which is an integer because of $\hat{f}_m, \hat{g}_n \in \mathbb{Z}$) of all coefficients $\hat{f}_i$ and $\hat{g}_j$.

We now come to the computation of $\hat{H}(x,y) = \gcd(\hat{F}, \hat{G})$. According to [13, 38], we can compute the subresultant sequence $\mathrm{Sres}_i(\hat{F}, \hat{G}; y) \in \mathbb{Z}[x,y]$ with $\tilde{O}(N^6 + N^5\mu)$ bit operations since the polynomials $\hat{F}$ and $\hat{G}$ have magnitude $(N, O(\mu + N \log N))$. The total degree of each polynomial $\mathrm{Sres}_i(\hat{F}, \hat{G}; y)$ is bounded by N^2, the y-degree is bounded by $N - i$, and all coefficients have bitsize $\tilde{O}(N(N + \mu))$. Let i_0 be the smallest index with $\mathrm{Sres}_{i_0}(\hat{F}, \hat{G}; y) \not\equiv 0$, then

$$\bar{H}(x,y) := \mathrm{Sres}_{i_0}(\hat{F}, \hat{G}; y) = \bar{h}_{i_0}(x) \cdot y^{i_0} + \cdots + \bar{h}_0(x) = \sum_{i,j} \bar{h}_{ij} \cdot x^i y^j$$

coincides with $\hat{H}$ up to a fraction $\frac{p(x)}{q(x)}$ with coprime $p, q \in \mathbb{Z}[x]$. That is,

$$\hat{H}(x,y) = \frac{p(x)}{q(x)} \cdot \bar{H}(x,y) = \left(\frac{p(x)}{q(x)} \cdot \bar{h}_{i_0}(x) \right) \cdot y^{i_0} + \cdots + \left(\frac{p(x)}{q(x)} \cdot \bar{h}_0(x) \right).$$

Again, we can assume that there exists no integer different from ± 1 that divides all coefficients $\bar{h}_{ij}$ of $\bar{h}$. Namely, we can divide $\bar{H}$ by the greatest common divisor of all $\bar{h}_{ij}$, and this computation uses $\tilde{O}(N^2 \cdot N(N + \mu))$ bit operations. Since the leading coefficients of $\hat{F}$ and $\hat{G}$ with respect to y are constants, the same also holds for $\hat{H} = \gcd(\hat{F}, \hat{G})$. Thus, p must be a constant and $q \simeq \bar{h}_{i_0}$. It follows that the primitive part[17] $\bar{h}^*_{i_0}$ of $\bar{h}_{i_0}$ divides all coefficients of $\bar{H}$, and that $\bar{H}/\bar{h}^*_{i_0} \simeq \hat{H}$. Hence, since $\hat{H}$ is primitive, we must have $\bar{H}/\bar{h}^*_{i_0} = \hat{H}$. The computation of $h^*_{i_0}$

[17] The primitive part P^* of a polynomial $P(x) = p_N \cdot x^N + \cdots + p_0 \in \mathbb{Z}[x]$ is defined as $P^*(x) := \gcd(p_N, \ldots, p_0)^{-1} \cdot P(x)$.

and $\bar{H}/\bar{h}_{i_0}^*$ needs $\tilde{O}(N^6+N^5\mu)$ bit operations, where we use the fact that the polynomials $\bar{h}_i(x)$ have magnitude $(N^2,\tilde{O}(N(N+\mu)))$ and that each division of $\bar{h}_i(x)$ by $h_{i_0}^*$ is remainder-free; cf. Lemma 12. Finally, computing $H=\gcd(F,G)$ from $\hat{H}=H(x+s_0\cdot y,y)$ uses $\tilde{O}(N^4+N^3\mu)$ bit operations since $\hat{H}$ has magnitude $(N,O(N\log N+N\cdot\log s_0+\mu))=(N,O(N\log N+\mu))$. □

We now come to the complexity analysis for TopNT. For the shearing step, we remark that there exist at most $n^{O(1)}$ many bad shearing factors s for which our algorithm does not succeed; see [4, Thm. 5] and [3, Prop. 11.23]. Thus, when choosing s at random, we can assume that we succeed for an integer s of bitsize $O(\log n)$. It follows that the sheared polynomial $f(x+sy,y)$ has magnitude $(n,O(\tau+n\log n))$. Hence, throughout the following considerations, we can assume that the leading coefficient $\operatorname{lcf}(f(x,y);y)$ of f (with respect to y) is an integer constant and that f has magnitude $(n,O(\tau+n\log n))$. We further define 2^t to be a power of two with $\operatorname{lcf}(f(x,y);y)\leq 2^t\leq 4\cdot\operatorname{lcf}(f(x,y);y)$.

Lemma 14. We can compute the entire subresultant sequence $\operatorname{Sres}_i(f,f_y;y)$, with $i=0,\ldots,n$, the polynomial $Q=\operatorname{res}(f_x^*,f_y^*;y)$, and the square-free parts R^* and Q^* of the corresponding polynomials $R=\operatorname{Sres}_0(f,f_y;y)=\operatorname{res}(f,f_y;y)$ and Q with $\tilde{O}(n^6+n^5\tau)$ bit operations. [18]

Proof. For two bivariate polynomials $g,h\in\mathbb{Z}[x,y]$ of magnitude (N,μ), computing the subresultant sequence $\operatorname{Sres}_i(g,h;y)\in\mathbb{Z}[x,y]$ together with the corresponding cofactor representations (i.e. the polynomials $u_i,v_i\in\mathbb{Z}[x,y]$ with $u_ig+v_ih=\operatorname{Sres}_i(g,h;y)$) needs $\tilde{O}(N^5\mu)$ bit operations [38, 13]. The total degree of the polynomials $\operatorname{Sres}_i(f,f_y;y)$ is bounded by N^2, the y-degree is bounded by $N-i$, and all coefficients have bitsize $\tilde{O}(N\mu)$. Furthermore, according to [**?** , §11.2], computing the square-free part of a univariate polynomial of magnitude (N,μ) uses $\tilde{O}(N^2\mu)$ bit operations, and the coefficients of the square-free part have bitsize $O(N+\mu)$. Hence, the claim concerning the computation of the polynomials $\operatorname{Sres}_i(f,f_y;y)$ and R^* follows from the fact that f and f_y have magnitude $(n,O(\tau+n\log n))$ and R has magnitude $(n^2,\tilde{O}(n^2+n\tau))$.

For the computation of the polynomials $f_x^*=\frac{f_x}{\gcd(f_x,f_y)}$ and $f_y^*=\frac{f_y}{\gcd(f_x,f_y)}$, Lemma 13 yields the bit complexity bound $\tilde{O}(n^6+n^5\tau)$. Lemma 11 implies that f_x^* and f_y^* have magnitude $(n,O(n+\tau))$, and thus computing Q needs $\tilde{O}(n^6+n^5\tau)$ bit operations. Q has magnitude $(n^2,\tilde{O}(n^2+n\tau))$. □

We now bound the cost for computing and comparing the roots of R and Q.

Lemma 15. The roots of the polynomials R and Q can be computed with $\tilde{O}(n^6+n^5\tau)$ bit operations. The same bound also applies to the number of bit operations that are needed to compute the multiplicities $\operatorname{mult}(\alpha,R)$ and $\operatorname{mult}(\alpha,Q)$, where α is a root of R.

Proof. According to Theorem 5, we can compute isolating disks for the roots of the polynomials R and Q together with the corresponding multiplicities with $\tilde{O}(n^6+n^5\tau)$ bit operations since R and Q have magnitude $(n^2,\tilde{O}(n^2+n\tau))$. For each root α of R, the algorithm returns a disk $\Delta^{(R)}(\alpha):=\Delta(\tilde{\alpha},r_\alpha)$ with radius $r_\alpha<\frac{\sigma(\alpha,R)}{64\deg R}$, and thus we can distinguish between real and non-real roots. A corresponding result also holds for each root β of Q, that is, each β is isolated by a

[18] The implementation from [4] does not compute the entire sub resultant sequence but only the resultants R^* and Q^*. This does not yield any improvement with respect to worst case bit complexity, however, a crucial speed up in practice can be observed.

disk $\Delta^{(Q)}(\beta)$ with radius less than $\frac{\sigma(\beta,Q)}{64\deg Q}$. Furthermore, for any given positive integer κ, we can further refine all isolating disks to a size of less than $2^{-\kappa}$ with $\tilde{O}(n^6+n^5\tau+n^2\kappa)$ bit operations.

For computing the multiplicities $\operatorname{mult}(\alpha,Q)$, where α is a root of R, we have to determine the common roots of R and Q. This can be achieved as follows: We first compute $d:=\deg\gcd(R^*,Q^*)$ for which we need $\tilde{O}(n^6+n^5\tau)$ bit operations. Namely, computing the gcd of two integer polynomials of magnitude (N,μ) needs $\tilde{O}(N^2\mu)$ bit operations. We conclude that R and Q have exactly d distinct roots in common. Hence, in the next step, we refine the isolating disks for R and Q until there are exactly d pairs $(\Delta^{(R)}(\alpha),\Delta^{(Q)}(\beta))$ of isolating disks that overlap. Since $P:=R\cdot Q$ has magnitude $(2n^2,\tilde{O}(n^2+n\tau))$, the minimal distance between two distinct roots α and β is bounded by the separation of P, thus it is bounded by $2^{-\tilde{O}(n^4+n^3\tau)}$. We conclude that it suffices to refine the isolating disks to a size of $2^{-\tilde{O}(n^4+n^3\tau)}$, hence the cost for the refinement is again bounded by $\tilde{O}(n^6+n^5)$. Now, for each of the d pairs $(\Delta^{(R)}(\alpha),\Delta^{(Q)}(\beta))$ of overlapping disks, we must have $\alpha=\beta$, and these are exactly the common roots of R and Q. □

From the above Lemma, we conclude that we can compute the numbers $k^+(\alpha)$ for all roots α of R with $\tilde{O}(n^6+n^5\tau)$ bit operations. Thus, the same bounds also applies to the computation of the upper bound $K^+=\sum_\alpha k^+(\alpha)$ for $K=\sum_\alpha k(\alpha)$.[19] For the computation of the lower bound K^-, we use the following result:

Lemma 16. We can compute K^- with $\tilde{O}(n^6+n^5\tau)$ bit operations.

Proof. From the proof of Lemma 14, we can assume that the leading coefficients $\mathrm{sr}_i\in\mathbb{Z}[x]$ of the subresultant sequence $\mathrm{Sres}_i(f,f_y;y)$ and the square-free part $S_0:=R^*$ of the resultant polynomial R are already computed. Note that all polynomials S_i and R_i as defined in (21) have coefficients of bitsize $\tilde{O}(n\tau+n^2)$ because all of them divide R^*. Thus, except for $(n\tau)^{O(1)}$ many bad primes, the modular computation over $\mathbb{Z}_p$ yields polynomials $S_i^{(p)},R_i^{(p)}\in\mathbb{Z}_p[x]$ with $\deg R_i^{(p)}=\deg R_i$ for all i, and thus $K^-=K$. Hence, we can assume that we only have to consider primes p of bitsize $O(\log(n\tau))$. Since we can compute the polynomials sr_i and R^* with $\tilde{O}(n^6+n^5\tau)$ bit operations, the same bound also applies to their modular computation over $\mathbb{Z}_p$.[20]

For the computation of the polynomials $S_i^{(p)}\in\mathbb{Z}_p[x]$, we have to perform at most n gcd computations (over $\mathbb{Z}_p$ with p of bit size $O(\log(n\tau))$) involving polynomials of degree n^2. Thus, the cost for these computations is bounded by $\tilde{O}(n\cdot n^2\log(n\tau))$ bit operations since computing the gcd of two polynomials in $\mathbb{Z}_p[x]$ of degree N can be achieved with $\tilde{O}(N)$ arithmetic operations in $\mathbb{Z}_p$ due to [51, Prop. 11.6]. For the computation of the $R_i^{(p)}$'s, we have to consider the cost for at most n (remainder-free) polynomial divisions. Again, for the latter computations, we need $\tilde{O}(n\cdot n^2\log(n\tau))$ bit operations. □

We remark that it is even possible to compute K directly in an *expected* number of bit operations bounded by $\tilde{O}(n^6+n^5\tau)$. Namely, following a randomized approach, the computation of the gcd of two integer polynomials of magnitude (N,μ) needs an expected number of bit operations bounded by $\tilde{O}(N^2+N\mu)$ according to [51, Prop. 11.11]. This yields the bound $\tilde{O}(n(n^4+n^3\tau))$ for the expected number of bit operations to compute the polynomials S_i from the subresultant

[19] For simplicity, we ignored that (in practice) K^+ can be computed much faster from the equality $K^+=n\cdot\deg R^*-\deg R+\deg\gcd(R^\infty,Q)$ instead of computing the $k^+(\alpha)$ first and, then, summing up all values.

[20] We remark that, in practice, we never compute the entire subresultant sequence over $\mathbb{Z}$. Here, we only assumed their exact computation in order to keep the argument simple and because of the fact that our overall complexity bound is not affected.

sequence $\mathrm{Sres}_i(f, f_y; y)$ and the polynomial $S_0 = R^*$. Obviously, the same bound also applies to the computation of the R_i's.

For the analysis of the curve topology algorithm, it remains to bound the cost for isolating the roots of the "fiber polynomials" $f_{\alpha_i}(y) \in \mathbb{R}[x]$ and $f_{\beta_i}(y) \in \mathbb{R}[x]$, where the α_i's are the real roots of R and the β_i's are arbitrary separating values in between. In practice, we recommend to choose arbitrary rational values β_i, however, following this straight forward approach yields a bit complexity of $\tilde{O}(n^7 + n^6\tau)$ for isolating the roots of the polynomials $f_{\beta_i}(y) \in \mathbb{Q}[y]$. Namely, if β_i is a rational value of bitsize L_i, then f_{β_i} has coefficients of bitsize $\tilde{O}(nL_i + \tau)$. Thus, isolating the roots of f_{β_i} needs $\tilde{O}(n^3L_i + n^2\tau)$ bit operations. However, since the separations of the α_i's are lower bounded by $2^{-\tilde{O}(n^4+n^3\tau)}$, we cannot get anything better than $\tilde{O}(n^4 + n^3\tau)$ for the largest L_i.

The crucial idea to improve upon the latter approach is to consider, for the values β_i, real roots of the polynomial $\hat{R}(x)$ instead, where

$$\hat{R} := \frac{(R^*)'}{\gcd((R^*)', (R^*)'')}$$

is defined as the square-free part of the derivative of R^*. Notice that the polynomials $\hat{R}$ and R do not share a common root. Furthermore, from the mean value theorem, we conclude that, for any two consecutive real roots of R, there exists a root of $\hat{R}$ in between these two roots. We can obtain such separating roots by computing isolating disks for all complex roots of $\hat{R}$ such that none of these disks intersects any of the isolating disks for the roots of R. The computation of $\hat{R}$ needs $\tilde{O}(n^6 + n^5\tau)$ bit operations since $(R^*)'$ has magnitude $(n^2, \tilde{O}(n^2 + n\tau))$. We can use the same argument as in the proof of Lemma 15 to show that it suffices to compute isolating disks for R and $\hat{R}$ of size $2^{-\tilde{O}(n^4+n^3\tau)}$ in order to guarantee that the disks do not overlap. Again, Theorem 4 shows that we achieve this with $\tilde{O}(n^6 + n^5\tau)$ bit operations.

Now, throughout the following considerations, we assume that the separating elements β_i are real roots of $\hat{R}$ with $\beta_{i-1} < \alpha_i < \beta_i$. We will show in Lemma 20 that, for isolating the roots of all polynomials f_{β_i} and f_{α_i}, we need only $\tilde{O}(n^6 + n^5\tau)$ bit operations. For this, we use the following result:

Lemma 17. Let $G \in \mathbb{Z}[x]$ be a polynomial of magnitude (N, μ). For an arbitrary subset $V' \subset \mathscr{V}(G)$, it holds that

$$\sum_{\alpha \in V'} \log \mathrm{Mea}(f_\alpha) = \tilde{O}(N\tau + n\mu + Nn), \text{ and } \sum_{\alpha \in V'} \tau_\alpha = \tilde{O}(N\tau + n\mu + Nn).$$

In particular, for $G \in \{R, \hat{R}\}$, the bound writes as $\tilde{O}(n^3 + n^2\tau)$.

Proof. The proof is almost identical to the proof of Lemma 5 in [29]. The only difference is that we consider a general G, whereas in [29], only the case $G = R$ has been treated. Note that $\mathrm{Mea}_\alpha \geq 1$ for every $\alpha \in V(G)$, and that the Mahler measure is multiplicative, that means, $\mathrm{Mea}(g)\,\mathrm{Mea}(h) = \mathrm{Mea}(gh)$ for arbitrary univariate polynomials g and h. Therefore,

$$\sum_{\alpha \in V'} \log \mathrm{Mea}(f_\alpha) \leq \sum_{\alpha \in \mathscr{V}(G)} \log \mathrm{Mea}(f_\alpha) = \log \mathrm{Mea}\left(\prod_{\alpha \in \mathscr{V}(G)} f_\alpha\right).$$

Considering f as a polynomial in x with coefficients in $\mathbb{Z}[y]$ yields

$$\prod_{\alpha \in \mathscr{V}(G)} f_\alpha = \frac{\mathrm{res}(f, G; x)}{\mathrm{lcf}(G)^n} \Rightarrow \sum_{\alpha \in V'} \log \mathrm{Mea}(f_\alpha) \leq \log \mathrm{Mea}(\mathrm{res}(f, G; x)).$$

It is left to bound the degree and the bitsize of $\operatorname{res}(f,G;x)$. Considering the Sylvester matrix of f and G (whose determinant defines $\operatorname{res}(f,G;x)$), we observe that it has n rows with coefficients of G (which are integers of size $O(\mu)$) and N rows with coefficients of f (which are univariate polynomials of magnitude $(n, O(\tau+n\log n))$). Therefore, the y-degree of $\operatorname{res}(f,G;y)$ is bounded by $O(nN)$, and its bitsize is bounded by $O(n(\mu+\log n)+N(\tau+n\log n)) = \tilde{O}(N\tau+n\mu+Nn)$. This shows that $\log \operatorname{Mea}(\operatorname{res}(f,G;x)) = \tilde{O}(N\tau+n\mu+Nn)$, and thus the first claim follows.

For the second claim, note that the absolute value of each coefficient of $f_\alpha(y)$ is bounded by $(n+1)\cdot\lambda M(\alpha)^n$, where $\lambda = 2^{O(\tau+n\log n)}$ is an upper bound for the absolute values of the coefficients of f. Thus, we have

$$\begin{aligned}\sum_{\alpha\in V'}\tau_\alpha \le \sum_{\alpha\in\mathscr{V}(G)}\tau_\alpha \le \sum_{\alpha\in\mathscr{V}(G)}\log((n+1)\lambda M(\alpha)^n)\\ = O(N(\tau+n\log n)+n\log\operatorname{Mea}(G) = \tilde{O}(N\tau+n\mu+Nn)\end{aligned}$$

For the last claim, note that, for $G\in\{R,\hat{R}\}$, we have $N\le n^2$ and $\mu=\tilde{O}(n^2+n\tau)$. □

Lemma 18. For $G\in\{R,\hat{R}\}$, we have

$$\sum_{\alpha\in V(G)}\sum_{i=1}^{k(\alpha)} m_{\alpha,i}\log M(\sigma_{\alpha,i}^{-1}) = \tilde{O}(n^4+n^3\tau) \quad\text{and}\quad \sum_{\alpha\in V(G)}\sum_{i=1}^{k(\alpha)}\log M(P_{\alpha,i}^{-1}) = \tilde{O}(n^4+n^3\tau).$$

Proof. First, consider $G=R$. For any root α of R, we define $m(\alpha) := \operatorname{mult}(\alpha,R)$. From (20), we conclude that $\sum_{i=1}^{k(\alpha)}(m_{\alpha,i}-1) = n-k(\alpha) \le n-k(\alpha)+\operatorname{mult}(Q,\alpha)\le\operatorname{mult}(\alpha,R)$. Furthermore, since $m(\alpha)\ge 1$, it follows that $m_{\alpha,i}\le m(\alpha)+1\le 2m(\alpha)$ for all i. Hence, we get

$$\begin{aligned}\sum_{\alpha\in V(R)}\sum_{i=1}^{k(\alpha)} m_{\alpha,i}\log M(\sigma_{\alpha,i}^{-1}) &\le \sum_{\alpha\in V(R)} 2\cdot m(\alpha)\cdot\sum_{i=1}^{k(\alpha)}\log M(\sigma_{\alpha,i}^{-1})\\ &\overset{(1)}{=}\tilde{O}\left(\sum_{\alpha\in V(R)} m(\alpha)\cdot\left(n\log\operatorname{Mea}(f_\alpha)+\log M(\operatorname{sr}_{n-k(\alpha)}(\alpha)^{-1})\right)\right)\\ &\overset{(2)}{=}\tilde{O}\left(\sum_{\alpha\in\mathscr{V}(R)} n\log\operatorname{Mea}(f_\alpha)+\log M(\operatorname{sr}_{n-k(\alpha)}(\alpha)^{-1})\right)\\ &\overset{(3)}{=}\tilde{O}(n^4+n^3\tau).\end{aligned}$$

For (1), we used [29, Thm. 9] to show that

$$\sum_{i=1}^{k(\alpha)}\log M(\sigma_{\alpha,i}^{-1}) = \tilde{O}(n\log\operatorname{Mea}(f_\alpha)+\log M(\operatorname{sr}_{n-k(\alpha)}(\alpha)^{-1}))\ . \tag{24}$$

(2) follows from the fact that each $\alpha\in V(R)$ occurs $m(\alpha)$ times in $\mathscr{V}(R)$. Finally, for (3), we apply Lemma 17 to bound the first sum and [29, Lemma 8] to bound the second one.

The second claim can be shown as follows. For each α, we first split the sum

$$\sum_{i=1}^{k(\alpha)}\log M(P_{\alpha,i}^{-1}) = \sum_{i=1}^{k(\alpha)}\log|P_{\alpha,i}|^{-1} + \sum_{i;|P_{\alpha,i}|>1}\log|P_{\alpha,i}|. \tag{25}$$

Then, for the first sum, we have

$$\begin{aligned}
\sum_{i=1}^{k(\alpha)} \log|P_{\alpha,i}|^{-1} &= \sum_{i=1}^{k(\alpha)} \log\prod_{j\neq i}|z_{\alpha,i}-z_{\alpha,j}|^{-m_{\alpha,j}} \\
&= \log\left(\prod_{i=1}^{k(\alpha)}\prod_{j\neq i}|z_{\alpha,i}-z_{\alpha,j}|\right)^{-1} + \sum_{i=1}^{k(\alpha)}\log\prod_{j\neq i}|z_{\alpha,i}-z_{\alpha,j}|^{-(m_{\alpha,j}-1)} \\
&\overset{(1)}{\leq} \log\frac{|\mathrm{lcf}(f_\alpha)^{2k(\alpha)-2}|\cdot\prod_{i=1}^{k(\alpha)} m_{\alpha,i}}{|\mathrm{sr}_{n-k(\alpha)}(\alpha)|} + \sum_{i=1}^{k(\alpha)}\sum_{j\neq i}\log\sigma_{\alpha,i}^{-(m_{\alpha,j}-1)} \\
&\overset{(2)}{=} \tilde{O}(n\tau) + \log|\mathrm{sr}_{n-k(\alpha)}(\alpha)|^{-1} + \sum_{i=1}^{k(\alpha)}\log\sigma_{\alpha,i}^{-1}\sum_{j\neq i}(m_{\alpha,j}-1) \\
&\overset{(3)}{\leq} \tilde{O}(n\tau) + \log|\mathrm{sr}_{n-k(\alpha)}|^{-1} + m(\alpha)\cdot\sum_{i=1}^{k(\alpha)}\log M(\sigma_{\alpha,i}^{-1}) \\
&\overset{(4)}{=} \tilde{O}(n\tau) + (m(\alpha)+1)\cdot\left(n\log\mathrm{Mea}\, f_\alpha + \log M(\mathrm{sr}_{n-k(\alpha)})^{-1}\right) \qquad (26)
\end{aligned}$$

For (1), we have rewritten the product as a subresultant term, where we used [3, Prop. 4.28]. Furthermore, the distances $|z_{\alpha,i}-z_{\alpha,j}|$ have been lower bounded by the separation of $z_{\alpha,i}$. For (2), note that lcf_α is an integer of bitsize $O(\tau+\log n)$, that $k\leq n$, and that $\prod_i m_{\alpha,i}\leq n^n$. For (3), we used that $\sum_{j=1}^{k(\alpha)}(m_{\alpha,j}-1)\leq m(\alpha)$, and, in (4), we applied (24). Now, summing up the expression in (26) over all $\alpha\in V(R)$ yields

$$\sum_{\alpha\in V(R)}\sum_{i=1}^{k(\alpha)}\log|P_{\alpha,i}|^{-1} = \tilde{O}(n^4+n^3\tau),$$

where we again use Lemma 17 and [29, Lemma 8].

For the second sum in (25), we use that (cf. proof of Theorem 5)

$$|P_{\alpha,i}| = \frac{|f_\alpha^{(m_{\alpha,i})}(z_{\alpha,i})|}{m_{\alpha,i}!\mathrm{lcf}(f_\alpha)} < n2^{\tau_\alpha+1}M(z_{\alpha,i})^n,$$

and thus

$$\begin{aligned}
\sum_{\alpha\in V(R)}\sum_{i;|P_{\alpha,i}|>1}\log|P_{\alpha,i}| &= \tilde{O}\left(\sum_{\alpha\in V(R)}\left(n\tau_\alpha + n\log\prod_{i=1}^{k(\alpha)}M(z_{\alpha,i})\right)\right) \\
&= \tilde{O}\left(\sum_{\alpha\in V(R)}(n\tau_\alpha + n\log\mathrm{Mea}(f_\alpha))\right) = \tilde{O}(n^4+n^3\tau)
\end{aligned}$$

according to Lemma 18. We conclude that

$$\sum_{\alpha\in V(R)}\sum_{i=1}^{k(\alpha)}\log M(P_{\alpha,i}^{-1}) = \tilde{O}(n^4+n^3\tau).$$

Now, consider the case $G=\hat{R}$. Note that, for each $\alpha\in V(\hat{R})$, we have $m_{\alpha,i}=1$ for all i, and thus $k(\alpha)=n$. Namely, R and $\hat{R}$ do not share a common root, and thus each polynomial f_α has only simple roots. Also, $V(\hat{R})=\mathcal{V}(\hat{R})$ since $\hat{R}$ is square-free. The following computation now shows the first claim

$$\sum_{\alpha\in V(\hat{R})}\sum_{i=1}^{k(\alpha)} m_{\alpha,i}\log M(\sigma_{\alpha,i}^{-1}) = \sum_{\alpha\in V(\hat{R})}\left(\sum_{i=1}^{n}\log M(\sigma_{\alpha,i}^{-1})\right) = \tilde{O}\left(\sum_{\alpha\in V(\hat{R})} n\log \mathrm{Mea}(f_\alpha)+\log M(\mathrm{sr}_0(\alpha)^{-1})\right)$$
$$=\tilde{O}\left(n^4+n^3\tau+\sum_{\alpha\in V(\hat{R})}\log M(R(\alpha)^{-1})\right).$$

In order to bound the sum in the above expression, note that

$$\sum_{\alpha\in V(\hat{R})}\log M(R(\alpha)^{-1}) = \sum_{\alpha\in V(\hat{R})}\log|R(\alpha)|^{-1}+\sum_{\alpha;|R(\alpha)|>1}|R(\alpha)|.$$

We first compute an upper bound for each value $|R(\alpha)|$. Since $R(x)$ has magnitude $(n^2,\tilde{O}(n\tau))$, it follows that $|R(\alpha)|$ has absolute value less than $2^{\tilde{O}(n\tau)}\cdot M(\alpha)^{n^2}$. Hence, for any subset $V'\subseteq V(\hat{R})$, it follows that

$$\sum_{\alpha\in V'}\log|R(\alpha)| \le \tilde{O}(n^3\tau)+n^2\log\mathrm{Mea}(\hat{R}) = \tilde{O}(n^4+n^3\tau).$$

Thus, it is left to show that $\sum_\alpha \log|R(\alpha)|^{-1} = \tilde{O}(n^4+n^3\tau)$, which follows from

$$\sum_{\alpha\in V(\hat{R})}\log|R(\alpha)|^{-1} = \log\prod_{\alpha\in V(\hat{R})}|R(\alpha)|^{-1} = \log\left(\frac{|\mathrm{lcf}(\hat{R})|^{\deg(R)}}{|\mathrm{res}(R,\hat{R})|}\right) = \tilde{O}(n^4+n^3\tau)\ . \qquad (27)$$

In the second equation we rewrote the product in terms of the resultant $\mathrm{res}(R,\hat{R})$ [3, Prop. 4.16]. Since R and $\hat{R}$ have no common root, we have $|\mathrm{res}(R,\hat{R})|\ge 1$. Thus, the last equation follows from the fact that the leading coefficient of $\hat{R}$ has bitsize $\tilde{O}(n^2+n\tau)$ and that $\deg(R)\le n^2$.

Similarly, for the second claim, we first derive an upper bound for $\sum_{\alpha\in V(\hat{R})}\sum_{i:P_{\alpha,i}>1}\log|P_{\alpha,i}|$. Again, we can use exactly the same argument as for the case $G=R$ to show that the latter sum is bounded by $\tilde{O}(n^4+n^3\tau)$. Hence, it suffices to prove that

$$\sum_{\alpha\in V(\hat{R})}\sum_{i=1}^{k(\alpha)}\log|P_{\alpha,i}|^{-1} = \tilde{O}(n^4+n^3\tau).$$

which follows from

$$\sum_{\alpha\in V(\hat{R})}\sum_{i=1}^{k(\alpha)}\log|P_{\alpha,i}|^{-1} = \sum_{\alpha\in V(\hat{R})}\sum_{i=1}^{n}-\log\prod_{j\ne i}|z_{\alpha,i}-z_{\alpha,j}| = \sum_{\alpha\in V(\hat{R})}\left(-\log\prod_{i=1}^{n}\prod_{j\ne i}|z_{\alpha,i}-z_{\alpha,j}|\right)$$
$$=\sum_{\alpha\in V(\hat{R})}\left(-\log\frac{|\mathrm{sr}_0(\alpha)|}{|\mathrm{lcf}(f_\alpha)^{2n-2}|}\right) = \sum_{\alpha\in V(\hat{R})}\left(-\log\frac{|R(\alpha)|}{|\mathrm{lcf}(f_\alpha)^{2n-2}|}\right) = \tilde{O}(n^4+n^3\tau)\ .$$

The last step follows from (27). We remark that the above computation is similar to the one for the case $G=R$. However, we used the fact that $\hat{R}$ is square-free, and thus all multiplicities $m_{\alpha,i}$ are equal to one. □

Lemma 19. Let $G\in\{R,\hat{R}\}$ and let $L_\alpha\in\mathbb{N}$ be arbitrary positive integers, where α runs over all real roots of G. Then, we can compute an absolute L_α approximations for all polynomials $f(\alpha,y)$ using

$$\tilde{O}(n^6+n^5\tau+n^2\sum_\alpha L_\alpha)$$

bit operations.

Proof. For each α, we use approximate interval arithmetic to compute an approximation of the polynomial f_α. If we choose a fixed point precision ρ, and a starting interval of size $2^{-\rho}$ that contains α, then the so-obtained interval approximation of f_α has interval coefficients of size $2^{-\rho+2}(n+1)^2 2^\tau M(\alpha)^n$; see again [28, Section 4] and [29, Section 5] for more details. Thus, in order to get an approximation of precision L_α of f_α, it suffices to consider a ρ of size $\tilde{O}(\tau + n\log M(\alpha) + L_\alpha)$. Thus, by doubling the precision ρ in each step, we eventually succeed for some $\rho = \rho_\alpha = \tilde{O}(\tau + n\log M(\alpha) + L_\alpha)$. The cost for the interval evaluations is then dominated (up to a logarithmic factor) by the cost in the last iteration. Thus, for a certain α, the cost is bounded by $\tilde{O}(n^2(\tau + n\log M(\alpha) + L_\alpha))$ since for each of the $n+1$ coefficients of f_α, we have to (approximately) evaluate an integer polynomial (i.e. the coefficients of f considered as a polynomial in y) of magnitude $(n, O(\tau + \log n))$ at $x = \alpha$. The total cost for all α is then bounded by

$$\tilde{O}\left(n^2 \cdot \sum_{\alpha\in\mathbb{R}\cap V(G)} \tau + n\log M(\alpha) + L_\alpha\right) = \tilde{O}(n^6 + n^5\tau + n^2 \sum_\alpha L_\alpha),$$

where we again used the result in Lemma 18. For the interval evaluations, we need an approximation of the root α to an absolute error of less than $2^{-\rho_\alpha}$. Such approximations are provided if we compute isolating disks of size less than $2^{-\kappa}$ for all roots of G, given that κ is larger than $\max_\alpha \rho_\alpha = \tilde{O}(\tau + n\max_\alpha \log M(\alpha) + \max_\alpha L_\alpha) = \tilde{O}(n^3 + n^2\tau + \max_\alpha L_\alpha)$. In the proof of Lemma 15, we have already shown that we can compute such disks using $\tilde{O}(n^6 + n^5\tau + n^2\kappa)$ bit operations. Thus, the claim follows. □

Lemma 20. Let $G \in \{R, \hat{R}\}$. Then, computing isolating disks for all roots of all f_α, $\alpha \in V(G) \cap \mathbb{R}$, together with the corresponding multiplicities uses

$$\tilde{O}(n^6 + n^5\tau)$$

bit operations.

Proof. For a fixed α, let B_α be the number of bit operations that are needed to compute isolating disks for all roots of f_α together with the corresponding multiplicities. We apply Theorem 3 to $2^{-t} \cdot f_\alpha$ which has exactly the same roots as f_α (remember that $\mathrm{lcf}(f(x,y);y) \in \mathbb{Z}$ and $\mathrm{lcf}(f(x,y);y) \le 2^t \le 4 \cdot \mathrm{lcf}(f(x,y);y)$). Then, we have

$$B_\alpha = \tilde{O}\left(n^3 + n^2\tau_\alpha + n \cdot \sum_{i=1}^{k(\alpha)} \left(m_{\alpha,i} \log M(\sigma_{\alpha,i}^{-1}) + \log M(P_{\alpha,i}^{-1})\right)\right).$$

The corresponding algorithm from Section 2.2 returns isolating disks for the roots $z_{\alpha,i}$ and their multiplicities $m_{\alpha,i}$. Furthermore, since the radius of the disk isolating $z_{\alpha,i}$ is smaller than $\sigma_{\alpha,i}/(64n)$, we can distinguish between real and non-real roots. The algorithm needs an absolute L_α-approximation of $2^{-t} \cdot f_\alpha$ (and thus an absolute $(L_\alpha - t)$-approximation of f_α) with

$$L_\alpha = \tilde{O}\left(n\tau_\alpha + \sum_{i=1}^{k(\alpha)} \left(m_{\alpha,i} \log M(\sigma_{\alpha,i}^{-1}) + \log M(P_{\alpha,i}^{-1})\right)\right).$$

From Lemma 19, we conclude that we can compute corresponding approximations for all f_α, $\alpha \in V(G) \cap \mathbb{R}$, with a number of bit operations bounded by

$$\tilde{O}(n^6 + n^5\tau + n^2 \cdot \sum_{\alpha\in V(G)\cap\mathbb{R}} L_\alpha).$$

The above expression is bounded by $\tilde{O}(n^6 + n^5\tau)$ because

$$\sum_{\alpha\in V(G)} \left(n\tau_\alpha + \sum_{i=1}^{k(\alpha)} \left(m_{\alpha,i} \log M(\sigma_{\alpha,i}^{-1}) + \log M(P_{\alpha,i}^{-1})\right)\right)$$

is bounded by $\tilde{O}(n^4+n^3\tau)$ according to Lemma 17 and 18. The same argument also shows that the sum over all B_α is even bounded by $\tilde{O}(n^5+n^4\tau)$. Hence, the claim follows. □

We can now formulate our main theorems of this section:

Theorem 6. Computing the topology of a real planar algebraic curve $C=V(f)$, where $f\in\mathbb{Z}[x,y]$ is a bivariate polynomial of total degree of n with integer coefficients of magnitude bounded by 2^τ, needs an expected number of bit operations bounded by

$$\tilde{O}(n^6+n^5\tau).$$

Proof. We already derived a bound of $\tilde{O}(n^6+n^5\tau)$ or better for each of the steps in the projection and in the lifting phase of our algorithm. The final connection phase is purely combinatorial since we ensure that each f_α, with α a root of the resultant R, has at most one multiple real root. Thus, we can compute all adjacencies in linear time with respect to the number of roots of critical and intermediate fiber polynomials. Since their number is bounded by $O(n^3)$, this step can be done in $O(n^3)$ operations. □

Notice that the problem of (real) solving a bivariate polynomial system $g(x,y)=h(x,y)=0$, with $g,h\in\mathbb{Z}[x,y]$ coprime polynomials, can be reduced to the problem of computing the topology of the planar algebraic curve $C:=\{(x,y)\in\mathbb{R}^2 : f(x,y)=0\}$, where $f:=g^2+h^2$. Namely, C coincides with the set of all points $(x,y)\in\mathbb{R}^2$ for which both polynomials g and h vanish. Since the degree of f is twice as large as the maximum of the degrees of the polynomials g and h, the following result follows in an almost straight forward manner: [21]

Theorem 7. Let $g,h\in\mathbb{Z}[x,y]$ be coprime polynomials of magnitude (n,τ). Then, we can compute isolating boxes for the real solutions of the system $g(x,y)=h(x,y)=0$ with an expected number of bit operations bounded by

$$\tilde{O}(n^6+n^5\tau).$$

Proof. As already mentioned above, the idea is to consider the polynomial $f(x,y):=g^2+h^2$ and to compute the topology of the curve $C:=C_\mathbb{R}$ defined by f. Since g and h are assumed to be coprime, the system $g=h=0$ has only finitely many solutions, and the set of these points coincides with the "curve" C. Hence, the topology algorithm returns a graph that consists of vertices only. According to Theorem 6, the cost the topology computation is bounded by $\tilde{O}(n^6+n^5\tau)$ bit operations in expectation since f has magnitude $(2n,O(\tau+\log n))$.

However, in general, our algorithm does not directly return the solutions of the initial system but the solutions of a sheared system $g(x+sy,y)=h(x+sy,y)=0$. Here, s is a positive integer of bitsize $O(\log n)$ for which TOPNT succeeds in computing the topology of the sheared curve $\hat{C}:=C_{s,\mathbb{R}}$ defined by $\hat{f}(x,y)=f(x+sy,y)=0$. Since $\hat{C}$ consists of isolated singular points only and there are no two covertical points (note that our algorithm only succeeds for an s for which there are no two covertical extremal points), it follows that, for each point $(\hat{x},\hat{y})\in\hat{C}$, $\hat{x}$ is a root of the resultant $\hat{R}=\operatorname{res}(\hat{f},\hat{f};y)$ and $\hat{y}$ is the unique (multiple) real root of $\hat{f}(\hat{x},y)$. The point $(\hat{x},\hat{y})$ is represented by an isolating box $B(\hat{x},\hat{y})=I(\hat{x})\times I(\hat{y})$, where $I(\hat{x})$ is the isolating interval for the root $\hat{x}$ of $\hat{R}$ and $I(\hat{y})$ is the isolating interval for the root $\hat{y}$ of $f(\hat{x},y)$. Each solution (x,y)

[21] We remark that we consider this result of rather theoretical interest because there exist efficient algorithms for solving bivariate systems that are comparable fast (in practice) as the fastest algorithms for topology computation; see [4, Section 6] for extensive benchmarks. Increasing the degree of the input polynomials by a factor of 2 certainly does not harm the asymptotic complexity bounds, however, it has a significant impact on the practical running times.

of the initial system can now be recovered from a unique solution $(\hat{x},\hat{y}) \in \hat{C}$. More precisely, $x = \hat{x} - s \cdot \hat{y}$ and $y = \hat{y}$. However, in order to obtain isolating boxes for the solutions (x,y), we have to refine the boxes $B(\hat{x},\hat{y})$ first such that the sheared boxes $B(x,y) := (I(\hat{x}) - s \cdot I(\hat{y}), I(\hat{y}))$ do not overlap. Note that the latter is guaranteed if both intervals $I(\hat{x})$ and $I(\hat{y})$ have width less than $\sigma(\hat{x},\hat{R})/(4|s|) \le \sigma(\hat{x},\hat{R})/4$. Namely, if the latter inequality holds, then the intervals $I(\hat{x}) - s \cdot I(\hat{y})$ are pairwise disjoint. Hence, it follows that the corresponding isolating intervals have to be refined to a width less than $w(\hat{x},\hat{y}) = \sigma(\hat{x},\hat{R})/n^{O(1)}$. For the resultant polynomial $\hat{R}$, we conclude from Theorem 5 that computing isolating intervals of size less $w(\hat{x},\hat{y})$ uses $\tilde{O}(n^6 + n^5\tau)$ bit operations since $\log M(w(\hat{x},\hat{y})^{-1}) = \tilde{O}(n^4 + n^3\tau)$ and $\hat{R}$ has magnitude $(n^2, \tilde{O}(n\tau))$. In order to compute an isolating interval of size $w(\hat{x},\hat{y})$ or less for the root $\hat{y}$ of $\hat{f}(\hat{x},y)$ (in fact, for all roots of $\hat{f}(\hat{x},y)$), we need

$$\tilde{O}(n^3 + n^2\tau_{\hat{x}} + n \cdot \textstyle\sum_{i=1}^{k(\hat{x})} \left(m_{\hat{x},i} \log M(\sigma_{\hat{x},i}^{-1}) + \log M(P_{\hat{x},i}^{-1}) \right) + (n \max_i m_{\hat{x},i}) \cdot \log M(w(\hat{x},\hat{y})^{-1})$$

bit operations; cf. the proof of Lemma 20 with $\alpha = \hat{x}$ and $f = \hat{f}$. Also, we need an approximation of precision $L_{\hat{x}}$ of $\hat{f}(\hat{x},y)$ with $L_{\hat{x}}$ bounded by

$$\tilde{O}(n\tau_{\hat{x}} + \textstyle\sum_{i=1}^{k(\hat{x})} \left(m_{\hat{x},i} \log M(\sigma_{\hat{x},i}^{-1}) + \log M(P_{\hat{x},i}^{-1}) \right) + (n \max_i m_{\hat{x},i}) \cdot \log M(w(\hat{x},\hat{y})^{-1}).$$

Since $\max_i m_{\hat{x},i} \le 2 \cdot \operatorname{mult}(\hat{x},\hat{R})$ and $w(\hat{x},\hat{y}) = \sigma(\hat{x},\hat{R})/n^{O(1)}$, it holds that

$$(n \max_i m_{\hat{x},i}) \cdot \log M(w(\hat{x},\hat{y})^{-1}) = \tilde{O}(n \cdot \operatorname{mult}(\hat{x},\hat{R}) \cdot \log M(\sigma(\hat{x},\hat{R})^{-1}).$$

Thus, summing up the cost for computing the roots of $\hat{f}(\hat{x},y)$ over all real roots of $\hat{R}$ yields the bound $\tilde{O}(n^5 + n^4\tau)$. Here, we use an analogous argument as in the proof of Lemma 20 and the fact that $\sum_{\hat{x}} n \cdot \operatorname{mult}(\hat{x},\hat{R}) \cdot \log M(\sigma(\hat{x},\hat{R})^{-1} = \tilde{O}(n^5 + n^4\tau)$. The more costly part is to compute the approximations of precision $L_{\hat{x}}$ of the polynomials $\hat{f}(\hat{x},y)$. Again, we can use Lemma 17 and 18 to show that $\sum_{\hat{x}} L_{\hat{x}} = \tilde{O}(n^4 + n^3\tau)$. Thus, from Lemma 19, we conclude that the approximations of the $\hat{f}(\hat{x},y)$'s can be computed with $\tilde{O}(n^6 + n^5\tau)$ bit operations. □

4. Conclusion

We presented an algorithm for isolating the roots of a complex univariate polynomial that can handle multiple roots provided the number k of distinct roots is part of the input and the coefficients can be approximated to an arbitrary precision. The algorithm uses approximate factorization as a subroutine. Any algorithm for approximate factorization that can be run with arbitrary precision can be used.

If used with Pan's algorithm [36] for approximate factorization, the algorithm is highly efficient:

- It solves the benchmark problem of isolating all roots of a polynomial p with *integer* coefficients of absolute value bounded by 2^τ with $\tilde{O}(n^3 + n^2\tau)$ bit operations. This matches the best bound known [17, Theorem 3.1].
- When combined with a a recent algorithm for computing the topology of a real planar algebraic curve specified as the zero set of a bivariate integer polynomial, it leads to improved complexity bounds for topology computation and and for isolating the real solutions of a bivariate polynomial system. For input polynomials of degree n and bitsize τ, we improve

the currently best running time from $\tilde{O}(n^9\tau + n^8\tau^2)$ (deterministic) to $\tilde{O}(n^6 + n^5\tau)$ (randomized) for topology computation and from $\tilde{O}(n^8 + n^7\tau)$ (deterministic) to $\tilde{O}(n^6 + n^5\tau)$ (randomized) for solving bivariate systems.

The considerable improvement of the bit complexity of the above problems related to the computation of a cylindrical algebraic decomposition mainly stems from the adaptivity of our root isolation method. That is, the precision demand as well as the number of bit operations to isolate the roots of the ”fiber polynomials” is directly related to the geometric locations of the corresponding roots. As a consequence, our analysis profits from amortization effects over the critical fibers. We expect that our adaptive complexity bound for root isolation will yield a a series of further complexity results for similar problems, where amortization effects take place.

A major open problem is whether there are deterministic algorithms for curve analysis and bivariate system solving of the same complexity.

References

[1] M. Agrawal, N. Kayal, and N. Saxena. PRIMES is in P. *Annals of Mathematics*, 160(2):pp. 781–793, 2004.

[2] D. S. Arnon, G. E. Collins, and S. McCallum. Cylindrical Algebraic Decomposition I. *SIAM Journal of Computing*, 13(4):865–889, 1984.

[3] S. Basu, R. Pollack, and M.-F. Roy. *Algorithms in Real Algebraic Geometry*. Springer, 2nd edition, 2006.

[4] E. Berberich, P. Emeliyanenko, A. Kobel, and M. Sagraloff. Exact Symbolic-Numeric Computation of Planar Algebraic Curves. *Theoretical Computer Science*, 491:1 –32, 2013.

[5] D. Bini and G. Fiorentino. Design, Analysis, and Implementation of a Multiprecision Polynomial Rootfinder. *Numerical Algorithms*, 23:127–173, 2000.

[6] Y. Bouzidi, S. Lazard, M. Pouget, and F. Rouillier. Rational univariate representations of bivariate systems and applications. In *ISSAC*, pages 109–116, 2013.

[7] Y. Bouzidi, S. Lazard, M. Pouget, and F. Rouillier. Separating linear forms for bivariate systems. In *ISSAC*, pages 117–124, 2013.

[8] M. Burr and F. Krahmer. Sqfreeeval: An (almost) optimal real-root isolation algorithm. *J. Symb. Comput.*, 47(2):153–166, 2012.

[9] J. Cheng, S. Lazard, L. Peñaranda, M. Pouget, F. Rouillier, and E. Tsigaridas. On the topology of real algebraic plane curves. *Mathematics in Computer Science*, 4(1):113–137, 2010.

[10] J.-S. Cheng, X.-S. Gao, and J. Li. Root isolation for bivariate polynomial systems with local generic position method. In *ISSAC*, pages 103–110, 2009.

[11] J.-S. Cheng, X.-S. Gao, and C.-K. Yap. Complete numerical isolation of real roots in zero-dimensional triangular systems. *Journal of Symbolic Computation*, 44(7):768 – 785, 2009.

[12] G. E. Collins, J. R. Johnson, and W. Krandick. Interval arithmetic in cylindrical algebraic decomposition. *J. Symb. Comput.*, 34(2):145–157, Aug. 2002.

[13] D. I. Diochnos, I. Z. Emiris, and E. P. Tsigaridas. On the Asymptotic and Practical Complexity of Solving Bivariate Systems Over the Reals. *J. Symb. Comput.*, 44(7):818–835, 2009.

[14] A. Eigenwillig, M. Kerber, and N. Wolpert. Fast and Exact Analysis of Real Algebraic Plane Curves. In *ISSAC 2007*, pages 151–158, New York, NY, USA, 2007. ACM.

[15] P. Emeliyanenko and M. Sagraloff. On the Complexity of Solving a Bivariate Polynomial System. In *ISSAC 2012*, pages 154–161, New York, NY, USA, 2012. ACM.

[16] I. Emiris and E. Tsigaridas. Real solving of bivariate polynomial systems. In *Computer Algebra in Scientific Computing*, volume 3718 of *Lecture Notes in Computer Science*, pages 150–161. 2005.

[17] I. Z. Emiris, V. Y. Pan, and E. P. Tsigaridas. Algebraic Algorithms. available at `tr.cs.gc.cuny.edu/tr/files/TR-2012001.pdf`, 2012.

[18] D. Eppstein, M. Paterson, and F. Yao. On Nearest-Neighbor Graphs. *Discrete & Comput. Geometry*, 17:263–282, 1997.

[19] M. Giusti, G. Lecerf, B. Salvy, and J.-C. Yakoubsohn. On location and approximation of clusters of zeros of analytic functions. *Foundations of Computational Mathematics*, 5(3):257–311, 2005.

[20] L. Gonzalez-Vega and M. E. Kahoui. An Improved Upper Complexity Bound for the Topology Computation of a Real Algebraic Plane Curve. *J. Complexity*, 12(4):527–544, 1996.

[21] L. González-Vega and M. E. Kahoui. An improved upper complexity bound for the topology computation of a real algebraic plane curve. *J. Complexity*, 12(4):527–544, 1996.

[22] X. Gourdon. *Combinatoire, Algorithmique et Géométrie des Polynomes.* PhD thesis, École Polytechnique, 1996.

[23] P. Henrici. *Elements of numerical analysis*. Wiley international edition. Wiley, 1964.

[24] P. Henrici. *Applied and computational complex analysis, Volume 1: Power series – integration – conformal mapping – location of zeros*. Wiley, 1974.

[25] P. Henrici. *Applied and Computational Complex Analysis, Power Series Integration Conformal Mapping Location of Zero*. Applied and Computational Complex Analysis. Wiley, 1988.

[26] H. Hong. An Efficient Method for Analyzing the Topology of Plane Real Algebraic Curves. *Mathematics and Computers in Simulation*, 42(4-6):571–582, 1996.

[27] J. R. Johnson. *Algorithms for Polynomial Real Root Isolation*. PhD thesis, The Ohio State University, 1991.

[28] M. Kerber and M. Sagraloff. Efficient Real Root Approximation. In *ISSAC 2011*, pages 209–216, New York, NY, USA, 2011. ACM.

[29] M. Kerber and M. Sagraloff. A Worst-case Bound for Topology Computation of Algebraic Curves. *J. Symb. Comput.*, 47(3):239–258, 2012.

[30] K.-I. Ko. Computational complexity of real functions. In *Complexity Theory of Real Functions*, Progress in Theoretical Computer Science, pages 40–70. Birkhäuser Boston, 1991.

[31] A. Kobel. Certified numerical root finding. Master's thesis, Fachbereich Informatik, University of Saarland, 2011.

[32] J. McNamee. *Numerical Methods for Roots of Polynomials*. Number 2 in Studies in Computational Mathematics. Elsevier Science, 2013.

[33] K. Mehlhorn and M. Sagraloff. A Deterministic Descartes Algorithm for Real Polynomials. *J. Symb. Comput.*, 46(1):70 – 90, 2011.

[34] K. Mehlhorn, M. Sagraloff, and P. Wang. From approximate factorization to root isolation. In *ISSAC 2013*, pages 283–290. ACM, 2013.

[35] M. Mignotte. *Mathematics for Computer Algebra*. Springer-Verlag, 1992.

[36] V. Pan. Univariate Polynomials: Nearly Optimal Algorithms for Numerical Factorization and Root Finding. *J. Symb. Comput.*, 33(5):701–733, 2002.

[37] Q. Rahman and G. Schmeisser. *Analytic Theory of Polynomials*. London Mathematical Society monographs. Clarendon Press, 2002.

[38] D. Reischert. Asymptotically Fast Computation of Subresultants. In *ISSAC 1997*, pages 233–240, New York, NY, USA, 1997. ACM.

[39] J. Renegar. On the worst-case arithmetic complexity of approximating zeros of polynomials. *Journal of Complexity*, 3(2):90 – 113, 1987.

[40] B. Rosser. Explicit bounds for some functions of prime numbers. *American Journal of Mathematics*, 63(1):pp. 211–232, 1941.

[41] F. Rouillier. On solving systems of bivariate polynomials. In K. Fukuda, J. Hoeven, M. Joswig, and N. Takayama, editors, *Mathematical Software ICMS 2010*, volume 6327 of *Lecture Notes in Computer Science*, pages 100–104. Springer Berlin Heidelberg, 2010.

[42] S. Rump. Ten methods to bound multiple roots of polynomials. *J. of Computation and Applied Mathematics (JCAM)*, pages 403–432, 2003.

[43] M. Sagraloff. When Newton meets Descartes: a simple and fast algorithm to isolate the real roots of a polynomial. In *ISSAC 2012*, pages 297–304. ACM, 2012.

[44] M. Sagraloff and C. K. Yap. A Simple but Exact and Efficient Algorithm for Complex Root Isolation. In *ISSAC 2011*, pages 353–360, New York, NY, USA, 2011. ACM.

[45] A. Schönhage. Asymptotically fast algorithms for the numerical multiplication and division of polynomials with complex coefficients. In J. Calmet, editor, *Computer Algebra*, volume 144, pages 3–15. Springer Berlin Heidelberg, 1982.

[46] A. Schönhage. The Fundamental Theorem of Algebra in Terms of Computational Complexity. Technical report, Math. Inst. Univ. Tübingen, 1982.

[47] A. Schönhage. Quasi-gcd computations. *Journal of Complexity*, 1(1):118 – 137, 1985.

[48] A. Sluis. Upperbounds for roots of polynomials. *Numerische Mathematik*, 15(3):250–262, 1970.

[49] A. Strzebonski. Cylindrical Algebraic Decomposition Using Validated Numerics. *J. Symb. Comp*, 41:1021–1038, 2006.

[50] A. Strzebonski and E. Tsigaridas. Univariate real root isolation in an extension field. ISSAC '11, pages 321–328, New York, NY, USA, 2011. ACM.

[51] J. von zur Gathen and J. Gerhard. *Modern Computer Algebra*. Cambridge University Press, Cambridge, 1999.

[52] J.-C. Yakoubsohn. Finding a cluster of zeros of univariate polynomials. *Journal of Complexity*, 16(3):603 – 638, 2000.

[53] C. Yap. *Fundamental Problems of Algorithmic Algebra*. Oxford University Press, 2000.

[54] C. Yap. In praise of numerical computation. In *Efficient Algorithms*, volume 5760 of *Lecture Notes in Computer Science*, pages 380–407. 2009.