Complexity Analysis of Root Clustering for a Complex Polynomial

Ruben Becker

MPI for Informatics, Saarbrücken Graduate School of Computer Science, Saarbrücken, Germany

Michael Sagraloff

MPI for Informatics, Saarbrücken, Germany

Vikram Sharma

Institute of Mathematical Sciences, Chennai, India

Juan Xu[1]

Beihang University, Beijing, China

Chee Yap[2]

Courant Institute of Mathematical Sciences, New York, USA

Abstract

Let $F(z)$ be an arbitrary complex polynomial. We introduce the **local root clustering problem**, to compute a set of natural ε-clusters of roots of $F(z)$ in some box region B_0 in the complex plane. This may be viewed as an extension of the classical root isolation problem. Our contribution is two-fold: we provide an efficient certified subdivision algorithm for this problem, and we provide a bit-complexity analysis based on the local geometry of the root clusters.

Our computational model assumes that arbitrarily good approximations of the coefficients of F are provided by means of an oracle at the cost of reading the coefficients. Our algorithmic techniques come from a companion paper Becker et al. (2017) and are based on the Pellet test, Graeffe and Newton iterations, and are independent of Schönhage's splitting circle method. Our algorithm is relatively simple and promises to be efficient in practice.

1. Introduction

The problem of computing the roots of a univariate polynomial F has a venerable history that dates back to antiquity. With the advent of modern computing, the subject received several newfound aspects McNamee and Pan (2013); Pan (1997); in particular, the introduction

Email addresses: ruben@mpi-inf.mpg.de (Ruben Becker), msagralo@mpi-inf.mpg.de (Michael Sagraloff), vikram@imsc.res.in (Vikram Sharma), xujuan0505@126.com (Juan Xu), yap@cs.nyu.edu (Chee Yap)

[1]Supported by China Scholarship Council, No.20150602005.

[2]Supported by NSF Grants #CCF-1423228 and #CCF-1564132.

of algorithmic rigor and complexity analysis has been extremely fruitful. This development is usually traced to Schönhage's 1982 landmark paper, "*Fundamental Theorem of Algebra in Terms of Computational Complexity*" Schönhage (1982). Algorithms in this tradition are usually described as "exact and efficient". Schönhage considers the problem of approximate polynomial factorization, that is, the computation of approximations $\tilde{z}_i$ of the roots z_i of F such that $\|F - \widetilde{F}\|_1 < 2^{-b} \cdot \|F\|_1$, where $\widetilde{F}(z) := \mathrm{lcf}(F) \cdot \prod_{i=1}^{n}(z - \widetilde{z}_i)$ and b is a given positive integer. The sharpest result for this problem is given by Pan (Pan, 2002, Theorem 2.1.1), (Pan, 1997, p.196). Hereafter, we refer to the underlying algorithm in this theorem as "Pan's algorithm". Under some mild assumption on F (i.e., $|z_i| \leq 1$ and $b \geq n \log n$), Pan's algorithm uses only $\tilde{O}(n \log b)$ arithmetic operations with a precision bounded by $\tilde{O}(b)$, and thus $\tilde{O}(nb)$ bit operations. This result further implies that the complexity of approximating all z_i's to any specified b/n bits, with $b > n \log n$, is also $\widetilde{O}(nb)$ (Pan, 2002, Corollary 2.1.2). Here, $\widetilde{O}$ means we ignore logarithmic factors in the displayed parameters. In a model of computation, where it is assumed that the coefficients of F are complex numbers for which approximations are given up to a demanded precision, the above bound is tight (up to poly-logarithmic factors) for polynomial factorization as well as for root approximation.

The preceding paragraph is concerned with *root approximation*, i.e., computing $\widetilde{z}_i$ such that $|\widetilde{z}_i - z_i| \leq \varepsilon$ for specified $\varepsilon > 0$. Our main focus is the stronger problem of *root isolations*, i.e., computing $(\widetilde{z}_i, r_i)$ such that $r_i \leq \varepsilon$ and the discs $\Delta(\widetilde{z}_i, r_i)$ centered at $\widetilde{z}_i$ of radius r_i are pairwise disjoint and contains z_i. A central focus in exact and efficient root approximation research has been to determine the complexity of *isolating* all the roots of an *integer* polynomial $F(z)$ of degree n with L-bit coefficients. We call this the **benchmark problem** in Sagraloff and Yap (2011) since this case is the main theoretical tool for comparing root isolation algorithms. Although this paper addresses complex root isolation, we will also refer to the related **real benchmark problem** which concerns real roots for integer polynomials.

Root isolation can be reduced to root approximate. Schönhage showed that, for a square-free polynomial F, it suffices to choose a b of size $\Omega(n(\log n + L))$ to ensure that the approximations $\widetilde{z}_i$ are isolated with 2ε taken as the root separation bound of F. Together with Pan's result on approximate polynomial factorization, this yields a complexity of $\widetilde{O}(n^2 L)$ for the benchmark problem. Interestingly, the latter bound was not explicitly stated until recently (Emiris et al., 2014, Theorem 3.1).

Mehlhorn et al. Mehlhorn et al. (2015) extend the latter result to (not necessarily square-free) polynomials F with arbitrary complex coefficients for which the number of distinct roots is given as an additional input. That is, Pan's algorithm is used as a blackbox with successively increasing precision b to isolate the roots of F. For the benchmark problem, this yields the bound $\widetilde{O}(n^3 + n^2 L)$; however, the actual cost adapts to the geometry of the roots, and for most input polynomials, the complexity is considerably lower than the worst case bound.

We further remark that it seems likely that the bound $\widetilde{O}(n^2 L)$ is also near-optimal for the benchmark problem because it is generally believed that Pan's algorithm is near-optimal for the problem of approximately factorizing a polynomial with complex coefficients. However, rigorous arguments for such claims are missing.

Until recently, it had been widely assumed that near-optimal bounds need the kind of "muscular" divide and conquer techniques such as the splitting circle method of Schönhage (which underlies most of the previous fast algorithms in the complexity literature). These algorithms are far from practical (see below). So, also the bound $\widetilde{O}(n^2(n + L))$ achieved by Mehlhorn et al. Mehlhorn et al. (2015) is mainly of theoretical interest as the algorithm uses Pan's method as a blackbox. Instead of these near-optimal algorithms, practitioners interested in a priori root iso-

lation invariably rely on subdivision methods. The classical example is real root isolation based on Sturm sequences (1829). For complex roots, Weyl (1924) introduced the quadtree method for Two types of subdivision algorithms are actively investigated currently: the **Descartes Method** Collins and Akritas (1976); Lane and Riesenfeld (1981); Rouillier and Zimmermann (2004); Schönhage (1982); Sagraloff (2012); Sagraloff and Mehlhorn (2015) and the **Evaluation Method** Burr et al. (2009); Burr and Krahmer (2012); Sharma and Yap (2012); Becker (2012); Sagraloff and Yap (2011); Kamath et al. (2011); Pan (2000). See Sagraloff and Yap (2011) for a comparison of Descartes and Evaluation (or Bolzano) methods.

The development of certain tools, such as the Mahler-Davenport root bounds Davenport (1985); Du et al. (2007), have been useful in deriving tight bounds on the subdivision tree size for certain subdivision algorithms Eigenwillig et al. (2006); Burr and Krahmer (2012); Sharma and Yap (2012). Moreover, most of these analyses can be unified under the "continuous amortization" framework Burr et al. (2009); Burr (2016) which can even incorporate bit-complexity. These algorithms only use bisection in their subdivision, which seems destined to lag behind the above "near optimal bounds" by a factor of n. To overcome this, we need to combine Newton iteration with bisection, an old idea that goes back to Dekker and Brent in the 1960s. In Pan (2000), Pan showed that theoretically, the near optimal bounds can be achieved with subdivision methods. In recent years, a formulation of Newton iteration due to Abbott Abbott (2014) and Sagraloff Sagraloff (2012) has proven especially useful. This has been adapted to achieve the recent near-optimal algorithms of Sagraloff and Mehlhorn Sagraloff (2012); Sagraloff and Mehlhorn (2015) for real roots, and Becker et al. (2017) for complex roots.

The Root Clustering Problem. In this paper, we are interested in root clustering. The requirements of root clustering represents a simultaneous strengthening of root approximation (i.e., the output discs must be disjoint) and weakening of root isolation (i.e., the output discs can have more than one root). Hereafter, "root finding" refers generally to any of the tasks of approximating, isolating or clustering roots.

For an analytic function $F : \mathbb{C} \to \mathbb{C}$ and a complex disc $\Delta \subseteq \mathbb{C}$, let $\mathcal{Z}(\Delta; F)$ denote the multiset of roots of F in Δ and $\#(\Delta; F)$ counts the size of this multiset. We write $\mathcal{Z}(\Delta)$ and $\#(\Delta)$ since F is usually supplied by the context. Any non-empty set of roots of the form $\mathcal{Z}(\Delta)$ is called a **cluster**. The disc Δ is called an **isolator** for F if $\#(\Delta) = \#(3\Delta) > 0$. Here, $k\Delta = k \cdot \Delta$ denotes the centrally scaled version of Δ by a factor $k \geq 0$. The set $\mathcal{Z}(\Delta)$ is called a **natural cluster** when Δ is an isolator. A set of n roots could contain $\Theta(n^3)$ clusters, but at most $2n - 1$ of these are natural. This follows from the fact that any two natural clusters are either disjoint or have a containment relationship. The benchmark problem is a global problem because it concerns *all* roots of the polynomial $F(z)$; we now address local problems where we are interested in finding only *some* roots of $F(z)$. For instance, Yakoubson Yakoubsohn (2000) gave a method to test if Newton iteration from a given point will converge to a cluster. In Yap et al. (2013), we introduced the following **local root clustering problem**: *given $F(z)$, a box $B_0 \subseteq \mathbb{C}$ and $\varepsilon > 0$, to compute a set $\{(\Delta_i, m_i) : i \in I\}$ where the Δ_i's are pairwise disjoint isolators, each of radius $\leq \varepsilon$ and $m_i = \#(\Delta_i) \geq 1$, such that*

$$\mathcal{Z}(B_0) \;\subseteq\; \bigcup\nolimits_{i \in I} \mathcal{Z}(\Delta_i) \;\subseteq\; \mathcal{Z}(2B_0).$$

We call the set $S = \{\Delta_i : i \in I\}$ (omitting the m_i's) a **solution** for the local root clustering instance $(F(z), B_0, \varepsilon)$. The roots in $2B_0 \setminus B_0$ are said to be **adventitious** because we are really only interested in roots in B_0. Suppose S and $\widehat{S}$ are both solutions for an instance $(F(z), B_0, \varepsilon)$. If $S \subseteq \widehat{S}$, then we call $\widehat{S}$ an augmentation of S. Thus any $\Delta \in \widehat{S} \setminus S$ contains only adventitious roots.

We solved the local root clustering problem in Yap et al. (2013) for any analytic function F, provided an upper on $\#(2B_0)$ is known, but no complexity analysis was given. Let us see why our formulation is reasonable. It is easy to modify our algorithm so that the adventitious roots in the output are contained in $(1+\delta)B_0$ for any fixed $\delta > 0$. We choose $\delta = 1$ for convenience. Some $\delta > 0$ is necessary because in our computational model where only approximate coefficients of F are available, we cannot decide the implicit "Zero Problem" Yap (2009) necessary to decide if the input has a root on the boundary of B_0, or to decide whether Δ contains a root of multiplicity $k > 1$. Thus, root clustering is the best one can hope for.

1.1. Main Result

In this paper, we describe a local root clustering algorithm and provide an analysis of its bit-complexity. Standard complexity bounds for root isolation are based on **synthetic parameters** such as degree n and bitsize L of the input polynomial. But our computational model for $F(z)$ has no notion of bit size. Moreover, to address "local" complexity of roots, we must invoke **geometric parameters** such as root separation Sagraloff (2012); Sagraloff and Mehlhorn (2015). We will now introduce new geometric parameters arising from cluster considerations.

Assume $F(z)$ has m distinct complex roots $z_1, \ldots, z_m$ where each z_j has multiplicity $n_j \geq 1$, thus $n = \sum_{j=1}^{m} n_j$ is the degree of $F(z)$. Let the magnitude of the leading coefficient of F be $\geq 1/4$, and the maximum coefficient magnitude $\|F\|_\infty$ be bounded by 2^{τ_F} for some τ_F.

Let k be the number of roots counted with multiplicities in $2B_0$. An input instance $(F(z), B_0, \varepsilon)$ is called **normal** if $k \geq 1$ and $\varepsilon \leq \min\left\{1, \frac{w_0}{96n}\right\}$ with w_0 the width of B_0. For any set $U \subseteq \mathbb{C}$, let $\overline{\log}(U) := \max(1, \log\sup(|z| : z \in U))$.

Our algorithm outputs a set of discs, each one contains a natural cluster. We provide a bit complexity bound of the algorithm in terms of the output.

Theorem A *Let S be the solution computed by our algorithm for a normal instance $(F(z), B_0, \varepsilon)$. Then there is an augmentation $\widehat{S} = \{D_i : i \in I\}$ of S such that the bit complexity of the algorithm is*

$$\widetilde{O}\Big(n^2 \overline{\log}(B_0) + n \sum\nolimits_{D\in\widehat{S}} L_D\Big) \tag{1}$$

with

$$L_D = \widetilde{O}\Big(\tau_F + n \cdot \overline{\log}(\xi_D) + k_D \cdot (k + \overline{\log}(\varepsilon^{-1})) + \overline{\log}(\prod\nolimits_{z_j\notin D} |\xi_D - z_j|^{-n_j})\Big) \tag{2}$$

where $k_D = \#(D)$, and ξ_D is an arbitrary root in D. Moreover, an L_D^*-bit approximation of the coefficients of F is required with $L_D^* := \max_{D\in\widehat{S}} L_D$.

The solution $\widehat{S}$ in this theorem is called an **augmented solution** for input $(F(z), B_0, \varepsilon)$. Each natural ε-cluster $D \in \widehat{S}$ is an isolator of radius $\leq \varepsilon$. From (1), we deduce:

Corollary to Theorem A
The bit complexity of the algorithm is bounded by

$$\widetilde{O}\Big(n^2(\tau_F + k + m) + nk\,\overline{\log}(\varepsilon^{-1}) + n\,\overline{\log}\,|\,\mathrm{GenDisc}(F_\varepsilon)|^{-1}\Big). \tag{3}$$

In case F is an integer polynomial, this bound becomes

$$\widetilde{O}\Big(n^2(\tau_F + k + m) + nk\,\overline{\log}(\varepsilon^{-1})\Big). \tag{4}$$

The bound (4) is the sum of two terms: the first is essentially the near-optimal root bound, the second is linear in k, n and $\overline{\log}(\varepsilon^{-1})$. This suggests that Theorem A is quite sharp.

On strong ε-clusters. Actually, the natural ε-clusters in the $\widehat{S}$ have some intrinsic property captured by the following definition. Two roots z, z' of F are **ε-equivalent**, written $z \overset{\varepsilon}{\sim} z'$, if there exists a disk $\Delta = \Delta(r, m)$ containing z and z' such that $r \leq \frac{\varepsilon}{12}$ and $\#(\Delta) = \#(114 \cdot \Delta)$. Clearly Δ is an isolator; from this, we see that ε-equivalence is an equivalence relationship. We define a **strong ε-cluster** to be any such ε-equivalence class. Unlike natural clusters, any two strong ε-clusters must be disjoint.

Theorem B
Each natural cluster $D \in \widehat{S}$ is a union of strong ε-clusters.

This implies that our algorithm will never split any strong ε-cluster. It might appear surprising that our "soft" techniques can avoid accidentally splitting a strong ε-cluster.

1.2. What is New

Our algorithm and analysis is noteworthy for its wide applicability: (1) We do not require square-free polynomials. This is important because we cannot compute the square-free part of $F(z)$ in our computational model where the coefficients of $F(z)$ are only arbitrarily approximated. Most of the recent fast subdivision algorithms for real roots Sagraloff (2012); Sagraloff and Mehlhorn (2015) require square-free polynomials. (2) We address the local root problem and provide a complexity analysis based on the local geometry of roots. Many practical applications (e.g., computational geometry) can exploit locality. The companion paper Becker et al. (2017) also gives a local analysis. However, it is under the condition that the initial box is not too large or is centered at the origin, and an additional preprocessing step is needed for the latter case. But our result does not depend on any assumptions on B_0 nor require any preprocessing. (3) Our complexity bound is based on cluster geometry instead of individual roots. To see its benefits, recall that the bit complexity in Becker et al. (2017) involves a term $\overline{\log}\,\sigma(z_i)^{-1}$ where $\sigma(z_i)$ is the distance to the nearest root of $F(z)$. If z_i is a multiple root, $\sigma(z_i) = 0$. If square-freeness is not assumed, we must replace $\sigma(z_i)$ by the distance $\sigma^*(z_i)$ to the closest root $\neq z_i$ (so $\sigma^*(z_i) > 0$). But in fact, our bound in (1) involves $T_D := \overline{\log} \prod_{z_j \notin D} |\xi_i - z_j|^{-n_j}$ which depends only on the inverse distance from a root within a cluster D to the other roots outside of D, which is smaller than $\overline{\log}\,\sigma^*(z_i)^{-1}$. So the closeness of roots within D has no consequence on T_D.

Why can't we just run the algorithm in Becker et al. (2017) by changing the stopping criteria so that it terminates as soon as a component C is verified to be a natural ε-cluster? Yes, indeed one can. But our previous method of charging the work associated with a box B to a root $\phi(B)$ may now cause a cluster of multiplicity k to be charged a total of $\Omega(k)$ times, instead of $\widetilde{O}(1)$ times. Cf. Lemma 11 below where $\phi(B)$ is directly charged to a cluster.

1.3. Practical Significance

Our algorithm is not only theoretically efficient, but has many potential applications. Local root isolation is useful in applications where the roots of interest lie in a known locality, and this local complexity can be much smaller than that of finding all roots. From this perspective, focusing on the benchmark problem is misleading for such applications.

We believe our algorithm is practical, and plan to implement it. Many recent subdivision algorithms were implemented, with promising results: Rouillier and Zimmermann (2004) engineered a very efficient Descartes method algorithm which is widely used in the Computer

Algebra community, through Maple. The CEVAL algorithm in Sagraloff and Yap (2011) was implemented in Kamath (2010); Kamath et al. (2011). Becker Becker (2012) gave a Maple implementation of the REVAL algorithm for isolating real roots of a square-free real polynomial. Most recently, Kobel, Rouillier and Sagraloff **?** implemented the ANewDsc algorithm from Sagraloff and Mehlhorn (2015), showing its all round superiority; it especially shines against known algorithms when roots are clustered.

Although there are several fast divide-and-conquer algorithms Renegar (1987); Neff and Reif (1996); Kirrinnis (1998), there is only one reported "proof of principle" implementation by Xavier ... [WIKI]. Pan notes (Pan, 2002, p. 703): "*Our algorithms are quite involved, and their implementation would require a non-trivial work, incorporating numerous known implementation techniques and tricks.*" Further (Pan, 2002, p. 705) "*since Schönhage (1982b) already has 72 pages and Kirrinnis (1998) has 67 pages, this ruled out a self-contained presentation of our root-finding algorithm*". Our paper Becker et al. (2017) is self-contained with over 50 pages, and explicit precision bounds for all numerical primitives: we use asymptotic bounds only in complexity analysis (since it has no consequence for implementations) but not in computational primitives.

2. Preliminary

We review the basic tools from Becker et al. (2017). The coefficients of F are viewed as an oracle from which we can request approximations to any desired absolute precision. Approximate complex numbers are represented by a pair of dyadic numbers, where the set of dyadic numbers (or BigFloats) may be denoted $\mathbb{Z}[\frac{1}{2}] := \{n2^m : n, m \in \mathbb{Z}\}$. We formalize[3] this as follows: a complex number $z \in \mathbb{C}$ is an **oracular number** if it is represented by an **oracle function** $\widetilde{z} : \mathbb{N} \to \mathbb{Z}[\frac{1}{2}]$ with some $\tau \geq 0$ such that for all $L \in \mathbb{N}$, $|\widetilde{z}(L) - z| \leq 2^{-L}$ and $\widetilde{z}(L)$ has $O(\tau + L)$ bits. The oracular number is said to be τ**-regular** in this case. In our computational model, the algorithm is charged the cost to read these $O(\tau + L)$ bits. This cost model is reasonable when z is an algebraic number because in this case, $\widetilde{z}(L)$ can be computed in time $\widetilde{O}(\tau + L)$ on a Turing machine. Following Becker et al. (2017); Yap et al. (2013), we can construct a procedure SoftCompare(z_ℓ, z_r) that takes two non-negative real oracular numbers z_ℓ and z_r with $z_\ell + z_r > 0$, that returns a value in $\{+1, 0, -1\}$ such that if SoftCompare(z_ℓ, z_r) returns 0 then $\frac{2}{3}z_\ell < z_r < \frac{3}{2}z_\ell$; otherwise SoftCompare$(z_\ell, z_r)$ returns sign$(z_\ell - z_r) \in \{+1, -1\}$. Note that SoftCompare is non-deterministic since its output depends on the underlying oracular functions used.

Lemma 1 (see (Becker et al., 2017, Lemma 4) and Yap et al. (2013)).
In evaluating SoftCompare(z_ℓ, z_r)*:*
(a) The absolute precision requested from the oracular numbers z_ℓ and z_r is at most

$$L = 2(\overline{\log}(\max(z_\ell, z_r)^{-1}) + 4).$$

(b) The time complexity of the evaluation is $\widetilde{O}(\tau + L)$ where z_ℓ, z_r are τ-regular.

The critical predicate for our algorithm is a test from Pellet (1881) (see Marden (1949)). Let $\Delta = \Delta(m, r)$ denote a disc with radius $r > 0$ centered at $m \in \mathbb{C}$. For $k = 0, 1, \ldots, n$ and $K \geq 1$,

[3] This is essentially the "bit-stream model", but the term is unfortunate because it suggests that we are getting successive bits of an infinite binary representation of a real number. We know from Computable Analysis that this representation of real numbers is not robust.

define the **Pellet test** $T_k(\Delta, K) = T_k(\Delta, K; F)$ as the predicate

$$|F_k(m)|r^k > K \cdot \sum_{i=0, i\neq k}^{n} |F_i(m)|r^i$$

Here $F_i(m)$ is defined as the Taylor coefficient $\frac{F^{(i)}(m)}{i!}$. Call the test $T_k(\Delta, K)$ a **success** if the predicate holds; else a **failure**. Pellet's theorem says that for $K \geq 1$, a success implies $\#(\Delta) = k$. Following Yap et al. (2013); Becker et al. (2017), we define the "soft version" of Pellet test $\widetilde{T}_k(\Delta)$ to mean that $\texttt{SoftCompare}(z_\ell, z_r) > 0$ where $z_\ell = |F_k(m)|r^k$ and $z_r = \sum_{i=0, i\neq k}^{n} |F_i(m)|r^i$. We need to derive quantitative information in case the soft Pellet test fails. Contra-positively, what quantitative information ensures that the soft Pellet test will succeed? Roughly, it is that $\#(\Delta) = \#(r\Delta) = k$ for a suitably large $r > 1$, as captured by the following theorem:

THEOREM 2.
Let k be an integer with $0 \leq k \leq n = \deg(F)$ and $K \geq 1$. Let $c_1 = 7kK$, and $\lambda_1 = 3K(n-k) \cdot \max\{1, 4k(n-k)\}$.
If $\#(\Delta) = \#(c_1\lambda_1\Delta) = k$, then

$$T_k(c_1\Delta, K, F) \quad \textit{holds.}$$

The factor $c_1\lambda_1$ is $O(n^4)$ in this theorem, an improvement from $O(n^5)$ in Becker et al. (2017). A proof is given in Appendix A. In application, we choose $K = \frac{3}{2}$ and thus $c_1 \cdot \lambda_1 \leq (7Kn) \cdot (12Kn^3) = 189n^4$. The preceding theorem implies that if $\#(\Delta) = \#(189n^4\Delta)$ then $T_k(\frac{21}{2}n\Delta, \frac{3}{2}, F)$ holds. This translates into the main form for our application:

COROLLARY
If $k = \#(\frac{1}{11}n\Delta) = \#(18n^3\Delta)$ then $T_k(\Delta, \frac{3}{2}; F)$ holds.

In other words, under the hypothesis of this Corollary, $\widetilde{T}_k(\Delta)$ succeeds. We need one final extension: instead of applying $\widetilde{T}_k(\Delta)$ directly on F, we apply $\widetilde{T}_k(\Delta(0,1))$ to the Nth Graeffe iterations of $F_\Delta(z) := F(m + rz)$. Here, $\Delta = \Delta(m, r)$ and $N = \lceil \log(1 + \log n) \rceil + 4 = O(\log\log n)$. The result is called the **Graeffe-Pellet test**, denoted $\widetilde{T}^G_k(\Delta) = \widetilde{T}^G_k(\Delta; F)$. As in Becker et al. (2017) we combine $\widetilde{T}^G_k(\Delta)$ for all $k = 0, 1, \ldots, n$ to obtain

$$\widetilde{T}^G_*(\Delta)$$

which returns the unique $k \in \{0, \ldots, n\}$ such that $\widetilde{T}^G_k(\Delta)$ succeeds, or else returns -1. We say that the test $\widetilde{T}^G_*(\Delta)$ **succeeds** iff $T^G_*(\Delta, K) \geq 0$.

The key property of $\widetilde{T}^G_i(\Delta)$ is (Becker et al., 2017, Lemma 6):

LEMMA 3 (Soft Graeffe-Pellet Test).
Let $\rho_1 = \frac{2\sqrt{2}}{3} \simeq 0.943$ and $\rho_2 = \frac{4}{3}$.
(a) If $\widetilde{T}^G_k(\Delta)$ succeeds then $\#(\Delta) = k$.
(b) If $\widetilde{T}^G_*(\Delta)$ fails then $\#(\rho_2\Delta) > \#(\rho_1\Delta)$.

The bit complexity of the combined test $\widetilde{T}^G_*(\Delta)$ is asymptotically the same as any individual test (Becker et al., 2017, Lemma 7):

LEMMA 4. *Let*

$$L(\Delta, F) := 2 \cdot (4 + \overline{\log}(\|F_\Delta\|_\infty^{-1})).$$

(a) To evaluate $\widetilde{T}_k^G(\Delta)$, it is sufficient to have an M-bit approximation of each coefficient of F where $M = \widetilde{O}(n\,\overline{\log}(m, r) + \tau_F + L(\Delta, F))$.
(b) The total bit-complexity of computing $\widetilde{T}_*^G(\Delta)$ is $\widetilde{O}(nM)$.

2.1. Box Subdivision

Let $A, B \subseteq \mathbb{C}$. Their **separation** is $\mathrm{Sep}(A, B) := \inf\{|a - b| : a \in A, b \in B\}$, and $\mathrm{rad}(A)$, the **radius** of A, is the smallest radius of a disc containing A. Also, ∂A denotes the boundary of A.

We use the terminology of subdivision trees (quadtrees) Becker et al. (2017). All boxes are closed subsets of $\mathbb{C}$ with square shape and axes-aligned. Let $B(m, w')$ denote the axes-aligned box centered at m of width $w(B) := w'$. As for discs, if $k \geq 0$ and $B = B(m, w')$, then kB denotes the box $B(m, kw')$. The smallest covering disc of $B(m, w')$ is $\Delta(m, \frac{1}{\sqrt{2}}w')$. If $B = B(m, w')$ then we define $\Delta(B)$ as the disc $\Delta(m, \frac{3}{4}w')$. Thus $\Delta(m, \frac{1}{\sqrt{2}}w')$ is properly contained in $\Delta(B)$. Any collection $\mathcal{S}$ of boxes is called a (box) **subdivision** if the interior of any two boxes in $\mathcal{S}$ are disjoint. The union $\bigcup \mathcal{S}$ of these boxes is called the **support** of S. Two boxes B, B' are **adjacent** if $B \cup B'$ is a connected set, equivalently, $B \cap B' \neq \emptyset$. A subdivision $\mathcal{S}$ is said to be **connected** if its support is connected. A **component** C is the support of some connected subdivision $\mathcal{S}$, i.e., $C = \bigcup \mathcal{S}$.

The **split** operation on a box B creates a subdivision $\mathtt{Split}(B) = \{B_1, \ldots, B_4\}$ of B comprising four congruent subboxes. Each B_i is a **child** of B, denoted $B \to B_i$. Therefore, starting from any box B_0, we may split B_0 and recursively split zero or more of its children. After a finite number of such splits, we obtain a **subdivision tree** rooted at B_0, denoted $\mathcal{T}_{subdiv}(B_0)$.

The **exclusion test** for a box $B(m, w')$ is $\widetilde{T}_0^G(\Delta(m, \frac{3w'}{4})) = \widetilde{T}_0^G(\Delta(B))$. We say that $B(m, w')$ is **excluded** if this test succeeds, and **included** if it fails. The key fact we use is a consequence of Lemma 3 for the test $\widetilde{T}_0^G(\Delta)$:

COROLLARY 5. *Consider any box $B = B(m, w')$.*
(a) If B is excluded, then $\#(\Delta(m, \frac{3w'}{4})) = 0$, so $\#(B) = 0$.
(b) If B is included, then $\#(\Delta(m, w')) > 0$, so $\#(2B) > 0$.

2.2. Component Tree

In traditional subdivision algorithms, we focus on the complexity analysis on the subdivision tree $\mathcal{T}_{subdiv}(B_0)$. But for our algorithm, it is more natural to work with a tree whose nodes are higher level entities called components above.

Typical of subdivision algorithms, our algorithm consists of several while loops, but for now, we only consider the main loop. This loop is controlled by the **active queue** Q_1. At the start of each loop iteration, there is a set of included boxes. The maximally connected sets in the union of these boxes constitute our (current) components. And the boxes in the subdivision of a component C are called the **constituent boxes** of C. While Q_1 is non-empty, we remove a component C from Q_1 for processing. There are 3 dispositions for C: We try to put C to the **output queue** Q_{out}. Failing this, we try a **Newton Step**. If successful, it produces a single new component $C' \subset C$ which is placed in Q_1. If Newton Step fails, we apply a **Bisection Step**. In this step, we split each constituent box of C, and apply the exclusion test to each of its four children. The set of included children are again organized into maximally connected sets $C_1, \ldots, C_t$ ($t \geq 1$). Each subcomponent C_i is either placed in Q_1 or Q_{dis}, depending on whether

C_i intersects the initial box B_0. The components in Q_{dis} are viewed as **discarded** because we do not process them further (but our analysis need to ensure that other components are sufficiently separated from them in the main loop). We will use the notation $C \to C'$ or $C \to C_i$ to indicate the parent-child relationship. The **component tree** is defined by this parent-child relationship, and denoted $\mathcal{T}_{comp}$. In Becker et al. (2017), the root of the component tree is B_0; we take $\frac{5}{4}B_0$ as the root to address boundary issues. So we write $\mathcal{T}_{comp} = \mathcal{T}_{comp}(\frac{5}{4}B_0)$ to indicate that $\frac{5}{4}B_0$ is the root. The leaves of $\mathcal{T}_{comp}$ are either discarded (adventitious) or output.

For efficiency, the set of boxes in the subdivision of a component C must maintain links to adjacent boxes within the subdivision; such links are easy to maintain because all the boxes in a component have the same width.

3. Component Properties

Before providing details about the algorithm, we discuss some critical data associated with each component C. Such data is subscripted by C. We also describe some qualitative properties so that the algorithm can be intuitively understood. Figure 1 may be an aid in the following description.

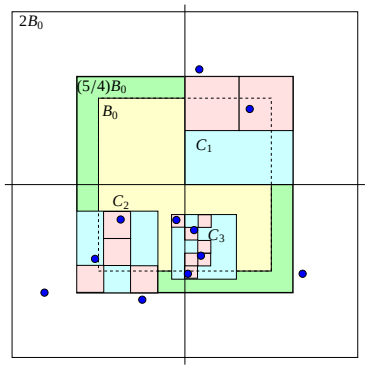

Figure 1: Three components C_1, C_2, C_3: blue dots indicate roots of F, pink boxes are constituent boxes, and the non-pink parts of each B_C is colored cyan. Only C_3 is confined.

(C1) All the constituent boxes of a component share a common width, denoted by w_C.

(C2) Our algorithm never discards any box B if B contains a root in B_0; it follows that all the roots in B_0 are contained in $\bigcup_C C$ where C ranges over components in $Q_0 \cup Q_1 \cup Q_{out}$ (at any moment during our algorithm).

(C3) Recall that a zero ζ of $F(z)$ in $2B_0 \setminus B_0$ is called adventitious. A component C is **adventitious** if $C \cap B_0$ is empty (placed in Q_{dis}). We say a component C is **confined** if $C \cap \partial(\frac{5}{4}B_0)$ is empty; otherwise it is non-confined. Figure 2 shows these different kinds of components. Note that after the preprocessing step, all components are confined.

(C4) If C, C' are distinct active components, then their separation $\mathrm{Sep}(C, C')$ is at least $\max\{w_C, w_{C'}\}$. If C is an adventitious component, then $\mathrm{Sep}(C, B_0) \geq w_C$. If C is a confined component, then $\mathrm{Sep}(C, \partial(\frac{5}{4}B_0)) \geq w_C$.

(C5) Let C^+ be the **extended component** defined as the set $\bigcup_{B\in\mathcal{S}_C} 2B$. If C and C' are distinct components, then C^+ and C'^+ are disjoint. Moreover, if C is confined, then $\#(C) = \#(C^+)$ (see Appendix B).

(C6) Define the **component box** B_C to be any smallest square containing C subject to $B_C \subseteq (5/4)B_0$. Define W_C as the width of B_C and the disc $\Delta_C := \Delta(B_C)$. Define R_C as the radius of Δ_C; note that $R_C = \frac{3}{4}W_C$.

(C7) Each component is associated with a "Newton speed" denoted by N_C with $N_C \geq 4$. A key idea in the Abbot- Sagraloff technique for Newton-Bisection is to automatically update N_C: if Newton fails, the children of C have speed $\max\left\{4, \sqrt{N_C}\right\}$ else they have speed N_C^2.

(C8) Let $k_C := \#(\Delta_C)$, the number of roots of $\mathcal{Z}(\Delta_C)$, *counted with multiplicity*. Note that k_C is not always available, but it is needed for the Newton step. We try to determine k_C before the Newton Step in the main loop.

(C9) A component C is **compact** if $W_C \leq 3w_C$. Such components have many nice properties, and we will require output components to be compact.

In recap, each component C is associated with the data:

$$w_C, W_C, M_C, B_C, \Delta_C, R_C, k_C, N_C.$$

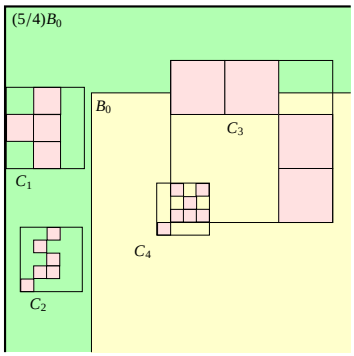

Figure 2: Four types of components: C_1 is not confined, the rest are confined; C_1 and C_2 are adventitious; C_3 may contain adventitious roots; C_4 has no adventitious roots.

4. The Clustering Algorithm

As outlined above, our clustering algorithm is a process for constructing and maintaining components, globally controlled by queues containing components. Each component C represents a non-empty set of roots. In addition to the queues Q_1, Q_{out}, Q_{dis} above, we also need a **preprocessing queue** Q_0. Furthermore, Q_1 is a priority queue such that the operation $C \leftarrow Q_1.pop()$ returns the component with the largest width W_C.

We first provide a high level description of the two main subroutines.

The **Newton Step** *Newton*(C) is directly taken from Becker et al. (2017). This procedure takes several arguments, *Newton*$(C, N_C, k_C,$
$x_C)$. The intent is to perform an order k_C Newton step:

$$x'_C \leftarrow x_C - k_C \frac{F(x_C)}{F'(x_C)}.$$

We then check whether $\mathcal{Z}(C)$ is actually contained in the small disc $\Delta' := \Delta(x'_C, r')$ where

$$r' := \max\{\varepsilon, w_C/(8N_C)\}. \tag{5}$$

This amounts to checking whether $\widetilde{T}^G_{k_C}(\Delta')$ succeeds. If it does, Newton test succeeds, and we return a new component C' that contains $\Delta' \cap C$ with speed $N_{C'} := (N_C)^2$ and constituent width $w_{C'} := \frac{w_C}{2N_C}$. The new component C' consists of at most 4 boxes and $W_{C'} \le 2w_{C'}$. In the original paper Becker et al. (2017), r' was simply set to $\frac{w_C}{8N_C}$; but (5) ensures that $r' \ge \varepsilon$. This avoids the overshot of Newton Step and simplifies our complexity analysis. If $\widetilde{T}^G_{k_C}(\Delta')$ fails, then Newton test fails, and it returns an empty set. In the following context, we simply denote this routine as "*Newton*(C)".

The **Bisection Step** *Bisect*(C) returns a set of components. Since it is different from that in Becker et al. (2017), we list the modified bisection algorithm in Figure 3.

We list the clustering algorithm in Figure 4.

Remarks on Root Clustering Algorithm:

1. The steps in this algorithm should appear well-motivated (after Becker et al. (2017)). The only non-obvious step is the test "$W_C \le 3w_C$" (colored in red).We may say C is **compact** if this condition holds. This part is only needed for the analysis; the correctness of the algorithm is not impacted if we simply replace this test by the Boolean constant `true` (i.e., allowing the output components to have $W_C > 3w_C$).
2. We ensure that $W_C \ge \varepsilon$ before we attempt to do the Newton Step. This is not essential, but simplifies the complexity analysis.

Based on the stated properties, we prove the correctness of our algorithm.

THEOREM 6 (Correctness). *The Root Clustering Algorithm halts and outputs a collection* $\{(\Delta_C, k_C) : C \in Q_{out}\}$ *of pairwise disjoint* ε*-isolators such that* $\mathcal{Z}(B_0) \subseteq \bigcup_{C \in Q_{out}} \mathcal{Z}(\Delta_C) \subseteq \mathcal{Z}(2B_0)$.

Proof. First we prove halting. By way of contradiction, assume $\mathcal{T}_{comp}$ has an infinite path $\frac{5}{4}B_0 = C_0 \to C_1 \to C_2 \to \cdots$. After $O(\log n)$ steps, the C_i's are in the main loop and satisfies $\#(C_i) = \#(C_i^+) \ge 1$. Thus the C_i converges to a point ξ which is a root of $F(z)$. For i large enough, C_i satisfies $W_{C_i} \le 3w_{C_i}$ and $w_{C_i} < \varepsilon$. Moreover, if C_i is small enough, $4\Delta_{C_i}$ will not intersect other components. Under all these conditions, the algorithm would have output such a C_i. This is a contradiction.

Upon halting, we have a set of output components. We need to prove that they represent a set of pairwise disjoint natural ε-clusters. Here, it is important to use the fact that Q_1 is a priority queue that returns components C in non-increasing width W_C. Suppose inductively, each component in the Q_{out} is represents a natural ε-cluster, and they are pairwise disjoint. Consider the next component C that we output: we know that $4\Delta_C$ does not intersect any components in $Q_1 \cup Q_{dis}$. But we also know that $C \cap 4\Delta_{C'} = \emptyset$ for any C' in Q_{out}. We claim that this implies that $3\Delta_C \cap C'$ must be empty. To see this, observe that $W_C \le W_{C'}$ because of the priority queue nature of Q_1. Draw the disc $4\Delta_{C'}$, and notice that the center of Δ_C cannot intersect $3\Delta_{C'}$. Therefore,

```
Bisect(C)
  OUTPUT: a set of components containing all
      the non-adventitious roots in C
      (but possibly some adventitious ones)
  Initialize a Union-Find data structure U
          for boxes.
  For each constituent box B of C
     For each child B' of B
        If (T̃_0^G(Δ(B')) fails)
           U.add(B')
           For each box B'' ∈ U adjacent to B'
              U.union(B', B'')
  Initialize Q to be empty.
  specialFlag ← true
  If (U has only one connected component)
     specialFlag ← false
  For each connected component C' of U
     If (C' intersects B_0) // C' not adventitious
        If (specialFlag)
           N_C' = 4
        Else
           N_C' = max{4, √N_C}
        Q.add(C')
     Else
        Q_dis.add(C')
  Return Q
```

Figure 3: Bisection Step

$3\Delta_C$ cannot intersect Δ'_C. This proves that C can be added to Q_{out} and preserve the inductive hypothesis.

It is easily verified that the roots represented by the confined components belong to $\frac{15}{8}B_0 \subset 2B_0$. But we must argue that we cover all the roots in B_0. How can boxes be discarded? They might be discarded in the Bisection Step because they succeed the exclusion test, or because they belong to an adventitious component. Or we might replace an entire component by a subcomponent in a Newton Step, but in this case, the subcomponent is verified to hold all the original roots. Thus, no roots in B_0 are lost. **Q.E.D.**

We now show some basic properties of the components produced in the algorithm.

LEMMA 7.
Let C be a component.
(a) If C is confined with $k = \#(C)$, then C has at most $9k$ constituent boxes. Moreover, $W_C \leq 3k \cdot w_C$.
(b) If $\mathcal{Z}(C)$ is strictly contained in a box of width w_C, then C is compact: $W_C \leq 3w_C$.

ROOT CLUSTERING ALGORITHM
Input: Polynomial $F(z)$, box $B_0 \subseteq \mathbb{C}$ and $\varepsilon > 0$
Output: Components in Q_{out} representing
natural ε-clusters of $F(z)$ in $2B_0$.

```
▷ Initialization
Q_out ← Q_1 ← Q_dis ← ∅.
Q_0 ← {(5/4)B_0} // initial component
▷ Preprocessing
While Q_0 is non-empty
   C ← Q_0.pop()
   If (C is confined and W_C ≤ w(B_0)/2)
         Q_1.add(C)
   Else
      Q_0.add(Bisect(C))
▷ Main Loop
While Q_1 is non-empty
   C ← Q_1.pop() // C has the largest W_C in Q_1
   If (4Δ_C ∩ C' = ∅ for all C' ∈ Q_1 ∪ Q_dis)
      k_C ← T̃*^G(Δ_C)
      If (k_C > 0) // Note: k_C ≠ 0.
         If (W_C ≥ ε)
            C' ← Newton(C)
            If (C' ≠ ∅)
               Q_1.add(C'); Continue
         Else if (W_C ≤ 3w_C) // C is compact
            Q_out.add(C); Continue
   Q_1.add(Bisect(C))
Return Q_out
```

Figure 4: Clustering Algorithm

(c) If there is a non-special path $(C_1 \to \cdots \to C)$ *where* C_1 *is special, then* $w_C \leq \frac{4w_{C_1}}{N_C}$.

Proof. Parts (a) and (b) are easy to verify. Part (c) is essentially from (Becker et al., 2017, Theorem 4) with a slight difference: we do not need to C_1 to be equal to the root $\frac{5}{4}B_0$. That is because our algorithm resets the Newton speed of the special component C_1 to 4. **Q.E.D.**

The next lemma addresses the question of lower bounds on the width w_C of boxes in components. If C is a leaf, then $w_C < \varepsilon$, but how much smaller than ε can it be? Moreover, we want to lower bound w_C as a function of ε.

LEMMA 8. *Denote* $k = \#(2B_0)$.
(a) If C *is a component in the pre-processing stage, then* $w_C \geq \frac{w(B_0)}{48k}$.
(b) Suppose $C_1 \to \cdots \to C_2$ *is a non-special path with* $W_{C_1} < \varepsilon$. *Then it holds*

$$\frac{w_{C_1}}{w_{C_2}} < 57k.$$

(c) Let C be a confined leaf in $\widehat{\mathcal{T}}_{comp}$ then

$$w_C > \frac{\varepsilon}{2}\Big(\frac{1}{114k}\Big)^k.$$

A proof of Lemma 8 is given in Appendix B.

We will need what we call the **small ε assumption**, namely, $\varepsilon \le \min\{1, w(B_0)/(96n)\}$. If this assumption fails, we can simply replace ε by $\varepsilon = \min\{1, w(B_0)/(96n)\}$ to get a valid bound from our analysis. This assumption is to ensure that no ε-cluster is split in the preprocessing stage.

5. Bound on Number of Boxes

In this section, we bound the number of boxes produced by our algorithm. All the proofs for this section are found in Appendix B.

The goal is to bound the number of all the constituent boxes of the components in $\mathcal{T}_{comp}$. But, in anticipation of the following complexity analysis, we want to consider an **augmented component tree** $\widehat{\mathcal{T}}_{comp}$ instead of $\mathcal{T}_{comp}$.

Let $\widehat{\mathcal{T}}_{comp}$ be the extension of $\mathcal{T}_{comp}$ in which, for each confined adventitious components in $\mathcal{T}_{comp}$, we (conceptually) continue to run our algorithm until they finally produce output components, i.e., leaves of $\widehat{\mathcal{T}}_{comp}$. As before, these leaves have at most 9 constituent boxes.

Since $C' \to C$ denote the parent-child relation, a path in $\mathcal{T}_{comp}$ may be written

$$P = (C_1 \to C_2 \to \cdots \to C_s). \tag{6}$$

We write w_i, R_i, N_i, etc, instead of $w_{C_i}, R_{C_i}, N_{C_i}$, etc.

A component C is **special** if C is the root or a leaf of $\widehat{\mathcal{T}}_{comp}$, or if $\#(C) < \#(C')$ with C' the parent of C in $\widehat{\mathcal{T}}_{comp}$; otherwise it is **non-special**. This is a slight variant of Becker et al. (2017).

We call P a **non-special path led by** C_1, if each C_i ($i = 2, \ldots, s$) is non-special, i.e., $\#(C_i) = \#(C_{i-1})$. The **special component tree** $\mathcal{T}^*_{comp}$ is obtained from $\widehat{\mathcal{T}}_{comp}$ by eliminating any non-special components while preserving the descendent/ancestor relationship among special nodes.

We now consider the length of an arbitrary non-special path as in (6). In (Becker et al., 2017, Lemma 10), it was shown that $s = O\Big(\log n + \log(\overline{\log}(w(B_0)) \cdot \overline{\log}(\sigma_F(2B)^{-1}))\Big)$. We provide an improved bound which is based on local data, namely, the ratio w_1/w_s only.

We define $s_{\max}$ to be the maximum length of a non-special path in $\widehat{\mathcal{T}}_{comp}$.

Theorem 9. *The length of the non-special path (6) satisfies*

$$s = O(\log\log\frac{w_1}{w_s} + \log n).$$

Particularly,

$$s_{\max} = O\Big(\log n + \log\log\frac{w(B_0)}{\varepsilon}\Big).$$

The proof of Theorem 9 is found in Appendix B.

Charging function $\phi_0(B)$. For each component C, define the **root radius** of C to be $r_C := \mathrm{rad}(\mathcal{Z}(C))$, that is the radius of the smallest disc enclosing all the roots in C. We are ready to define a charging function ϕ_0 for each box B in the components of $\widehat{\mathcal{T}}_{comp}$: Let $C_B \in \widehat{\mathcal{T}}_{comp}$ be the component

of which B is a constituent box. Let ξ_B be any root in $2B$. There are two cases: (i) If C_B is a confined component, there is a unique maximum path in $\widehat{\mathcal{T}}_{comp}$ from C_B to a confined leaf E_B in $\widehat{\mathcal{T}}_{comp}$ containing ξ_B. Define $\phi_0(B)$ to be the first special component C along this path such that

$$r_C < 3w_B. \tag{7}$$

where w_B is the width of B. (ii) If C_B is not confined, it means that C is a component in the preprocessing stage. In this case, define $\phi_0(B)$ to be the largest natural ε-cluster containing ξ_B. Notice that $\phi_0(B)$ is a special component in (i) but a cluster in (ii).

LEMMA 10. *The map ϕ_0 is well-defined.*

Proof. Consider the component C_B of which B is a constituent box. There are two cases in our definition of ϕ_0:

(i) If C_B is a confined component, it is easy to see that we can find a root $\xi_B \in 2B$, and fix a unique maximum path in $\widehat{\mathcal{T}}_{comp}$ from C_B to a confined leaf E_B in $\widehat{\mathcal{T}}_{comp}$ containing ξ_B. It suffices to prove that we can always find a special component C in this path such that $r_C < 3w_B$. This is true because $r_{E_B} < 3w_{E_B}$; to see this, note that E_B is a confined leaf of $\widehat{\mathcal{T}}_{comp}$. Thus $W_{E_B} \le 3w_{E_B}$ (this is the condition for output in the main loop of the Root Clustering Algorithm). It follows $r_{E_B} \le \frac{\sqrt{2}}{2} \cdot 3w_{E_B} < 3w_{E_B}$. Hence $r_{E_B} < 3w_{E_B} < 3w_B$. we can always find a first special component along the path from C_B to E_B such that (7) is satisfied.

(ii) If C_B is a non confined component, we can also find a root ξ_B in $2B$, and we can always charge B to the largest natural ε-cluster containing ξ_B. **Q.E.D.**

Using this map, we can now bound the number of boxes.

LEMMA 11. *The total number of boxes in all the components in $\widehat{\mathcal{T}}_{comp}$ is*

$$O(t \cdot s_{\max}) = O(\#(2B_0) \cdot s_{\max})$$

with $t = |\{\phi_0(B) : B \text{ is any box in } \widehat{\mathcal{T}}_{comp}\}|$.

This improves the bound in Becker et al. (2017) by a factor of $\log n$. A proof for Lemma 11 is found in Appendix B.

6. Bit Complexity

Our goal is to prove the bit-complexity theorem stated in the Introduction. From the discussion in (Becker et al., 2017, Theorem 7), the total cost of all the $\widetilde{T}^G$ tests is the main cost of the whole algorithm. Thus we need to account for the cost of $\widetilde{T}^G$ tests on all the concerned boxes and components.

The road map is as follows: we will charge the work of each box B (resp., component C) to some natural ε-cluster denoted $\phi(X)$ (resp., $\phi(C)$). We show that each cluster $\phi(X)$ (X is a box or a component) is charged $\widetilde{O}(1)$ times. Summing up over these clusters, we obtain our bound.

We may assume $\overline{\log}(B_0) = O(\tau_F)$ since Cauchy's root bound implies that any root z_i satisfies $|z_i| \le 1 + 4 \cdot 2^\tau$, thus we can replace B_0 by $B_0 \cap B(0, 2 + 8 \cdot 2^\tau)$.

Cost of $\widetilde{T}^G$-tests and Charging function $\phi(X)$: Our algorithm performs 3 kinds of $\widetilde{T}^G$-tests:

$$\widetilde{T}^G_*(\Delta_C), \quad \widetilde{T}^G_{k_C}(\Delta'), \quad \widetilde{T}^G_0(\Delta(B)) \tag{8}$$

respectively appearing in the main loop, the Newton Step and the Bisection Step. We define the **cost** of processing component C to be the costs in doing the first 2 tests in (8), and the **cost** of processing a box B to be the cost of doing the last test. Note that the first 2 tests do not apply to the non-confined components (which appear in the preprocessing stage only), so there is no corresponding cost.

We next "charge" the above costs to natural ε-clusters. More precisely, if X is a confined component or any box produced in the algorithm, we will charge its cost to a natural ε-cluster denoted $\phi(X)$: (a) For a special component C, let $\phi(C)$ be the natural ε-cluster $\mathcal{Z}(C')$ where C' is the confined leaf of $\mathcal{T}^*_{comp}$ below C which minimizes the length of path from C to C' in $\mathcal{T}^*_{comp}$. (b) For a non-special component C, we define $\phi(C)$ to be equal to $\phi(C')$ where C' is the first special component below C. (c) For a box B, we had previously defined $\phi_0(B)$ (see Section 5). There are two possibilities: If $\phi_0(B)$ is defined as a special component, then $\phi(\phi_0(B))$ was already defined in (a) above, so we let $\phi(B) := \phi(\phi_0(B))$. Otherwise, $\phi_0(B)$ is defined as a natural ε-cluster, and we let $\phi(B) = \phi_0(B)$.

LEMMA 12. *The map ϕ is well-defined.*

Proof. For a special component C, to define $\phi(C)$ we first consider C', defined as the confined leaf such that path $(C \to \cdots \to C')$ is the shortest in $\mathcal{T}^*_{comp}$. This path has length at most $\log n$ since there exists a path of length at most $\log n$ in which we choose the special node with the least $\#(C_i)$ at each branching (this was the path chosen in Becker et al. (2017)). Hence, $\phi(C)$ is well-defined. The map ϕ for a non-special component and a box are defined based on that for a special component, it is easy to check that they are well-defined.

It remains to prove that in the case where $\phi_0(B)$ is a natural ε-cluster, the map ϕ is well-defined. This follows from Lemma 10. **Q.E.D.**

Define $\widehat{S}$ to be the range of ϕ, so it is a set of natural ε- clusters. The clusters in $\widehat{S}$ are of two types: those defined by the confined leaves of $\widehat{\mathcal{T}}_{comp}$, and those largest ε-clusters of the form $\phi(B)$ with B in non-confined components.

We use the notation $\widetilde{O}(1)$ to refer to a quantity that is $O((\underline{\log}(n\tau \log(\varepsilon^{-1})))^i)$ for some constant i. To indicate the complexity parameters explicitly, we could have written "$\widetilde{O}_{n,\tau,\underline{\log}(\varepsilon^{-1})}(1)$".

LEMMA 13. *Each natural ε-cluster in $\widehat{S}$ is charged $O(s_{\max} \log n)$ times, i.e., $\widetilde{O}(1)$ times.*

Proof. First consider the number of components mapped to a same natural ε-cluster. From the definition of $\phi(C)$ for a special component, it is easy to see that the number of special components mapped to a same natural ε-cluster is at most $\log n$. Thus the number of non-special components mapped to a same natural ε-cluster is bounded by $O(s_{\max} \log n)$. Hence the number of components mapped to a same natural ε-cluster is bounded by $O(s_{\max} \log n)$.

Then we consider the number of boxes mapped to a same natural ε-cluster. By Lemma 11, the number of boxes charged to a same special component by ϕ_0 is bounded by $O(s_{\max})$, and the number of special components mapped to a same natural ε-cluster is bounded by $O(\log n)$, thus the number of boxes mapped to a same natural ε-cluster is bounded by $O(s_{\max} \log n) = \widetilde{O}(1)$. Also by Lemma 11, the number of boxes charged to a same natural ε-cluster by ϕ_0 is bounded by $O(\log n)\widetilde{O}(1)$.

In summary, each natural ε-cluster is mapped $O(s_{\max} \log n) = \widetilde{O}(1)$ times. **Q.E.D.**

Based on the charging map ϕ, we can derive a bound for the cost of processing each component and box.

LEMMA 14. *Denote $k = \#(2B_0)$.*
(a) Let B be a box produced in the algorithm. The cost of processing B is bounded by

$$\widetilde{O}\Big(n \cdot [\tau_F + n\,\overline{\log}(B) + k_D \cdot (\overline{\log}(\varepsilon^{-1}) + k) + T_D]\Big) \tag{9}$$

with $D = \phi(B)$, $k_D = \#(D)$ and

$$T_D := \overline{\log} \prod_{z_j \notin D} |\xi_D - z_j|^{-n_j}. \tag{10}$$

where ξ_D is an arbitrary root contained in D.
(b) Let C be a component produced in the main-loop, and let C_0 be the last special component above C, then the cost of processing a component C is bounded by

$$\widetilde{O}\Big(n \cdot [\tau_F + n\,\overline{\log}(C) + n\,\overline{\log}(w_{C_0}) + k_D \cdot (\overline{\log}(\varepsilon^{-1}) + k) + T_D]\Big) \tag{11}$$

where D is an arbitrary cluster contained in C, $k_D = \#(D)$ and T_D is as defined in (C.2).

A proof of Lemma 14 is found in Appendix C.

We are almost ready to prove the theorems announced in Section 1.1. Theorem A is easier to prove if we assume that the initial box B_0 is **nice** in the following sense:

$$\max_{z \in 2B_0} \overline{\log}(z) = O(\min_{z \in 2B_0} \overline{\log}(z)). \tag{12}$$

Here we only prove the case where the initial box is nice, and a complete proof of Theorem A is provided in the end of Appendix C.

In the nice case, the following lemma bounds the cost of processing X where X is a box or a component.

LEMMA 15. *If the initial box is nice, the cost of processing X (where X is a box or a component) is bounded by*

$$\widetilde{O}(n \cdot L_D)$$

bit operations with $D = \phi(X)$ and with L_D defined in (2). Moreover, an L_D-bit approximation of F is required.

Proof. Note that if the initial box satisfies (12), then it holds that $\overline{\log}(B) = O(\overline{\log}(\xi))$ and $\overline{\log}(C) = O(\overline{\log}(\xi))$ for any box B and component C and any root $\xi \in 2B_0$. And we know that $\phi(C) \subset C$. **Q.E.D.**

Thus this Lemma is a direct result form Lemma 14. Using this lemma, we could prove Theorem A of Section 1.1 under the assumption that B_0 is nice.

Before we prove the Theorem A in Section 1.1, we want to address a trivial case excluded by the statement in that theorem. In Theorem A, we assumed that the number of roots k in $2B_0$ is at least 1. If $k = 0$, then the algorithm makes only one test, $\widetilde{T}_0^G(\frac{5}{4}B_0)$. We want to bound the complexity of this test. Denoting the center of B_0 as M_0, the distance from M_0 to any root is at least $\frac{w(B_0)}{2}$. Thus $|F(M_0)| > |\operatorname{lcf}(F)| \cdot (\frac{w(B_0)}{2})^n$. Thus by (Becker et al., 2017, Lemma 7), the cost of this $\widetilde{T}_k^G$ test is bounded by $\widetilde{O}\Big(n\tau_F + n^2\,\overline{\log}(B_0) + n\,\overline{\log}(w(B_0)^{-1})\Big)$. Now we return to the Theorem A in the introduction.

Theorem A *Let S be the solution computed by our algorithm for a normal instance $(F(z), B_0, \varepsilon)$. Then there is an augmentation $\widehat{S} = \{D_i : i \in I\}$ of S such that the bit complexity of the algorithm*

is

$$\widetilde{O}\Big(n\sum\nolimits_{D\in\widehat{S}} L_D\Big)$$

with

$$L_D = \widetilde{O}\Big(\tau_F + n\cdot \overline{\log}(\xi_D) + k_D\cdot(\overline{\log}(k+\varepsilon^{-1})) + \overline{\log}(\prod\nolimits_{z_j\notin D}|\xi_D - z_j|^{-n_j})\Big)$$

where $k_D = \#(D)$, and ξ_D is an arbitrary root in D. Moreover, an L_D^*-bit approximation of the coefficients of F is required with $L_D^* := \max_{D\in\widehat{S}} L_D$.

The set $\widehat{S}$ in this theorem is precisely the range of our charge function ϕ, as defined in the text.

Lemma 16. *If B_0 satisfies (12), then the Theorem A holds.*

Proof. Recall that the number of components and that of boxes mapped to any natural ε-cluster is bounded by $\log n\cdot s_{\max}$. Thus from Lemma 15, the cost of processing all the components and boxes mapped to a natural cluster $D\in\widehat{S}$ is bounded by $\widetilde{O}(\log n\cdot s_{\max}\cdot nL_D)$. But $\log n\cdot s_{\max}$ is negligible in the sense of being $\widetilde{O}(1)$. Thus the total cost of all the $\widetilde{T}^G$ tests in the algorithm can be bounded by

$$\widetilde{O}\Big(n\sum\nolimits_{D\in\widehat{S}} L_D\Big)$$

with L_D defined in (2) and $\widehat{S}$ is the range of ϕ. And it is easy to see that $\widetilde{O}\Big(n\sum_{D\in\widehat{S}} L_D\Big)$ is bounded by (1).

There is another issue concerning total cost (as in (Becker et al., 2017, Theorem 7)): There is a non-constant complexity operation in the main loop: in each iteration, we check if $4\Delta_C\cap C'$ is empty. This cost is $O(n)$ since C' has at most $9n$ boxes. This $O(n)$ is already bounded by the cost of the iteration, and so may be ignored. **Q.E.D.**

The appendix will prove Theorem A holds even if B_0 is not nice.

In Becker et al. (2017), the complexity bound for global root isolation is reduced to the case where B_0 is centered at the origin. This requires a global pre-processing step. It is unclear that we can adapt that pre-processing to our local complexity analysis.

The bit complexity in Theorem A is based on geometric parameters, we can also write it in terms of synthetic parameters, although the the latter bound is not as sharper as the former one.

Corollary to Theorem A
The bit complexity of the algorithm is bounded by

$$\widetilde{O}\Big(n^2(\tau_F + k + m) + nk\,\overline{\log}(\varepsilon^{-1}) + n\,\overline{\log}\,|\operatorname{GenDisc}(F_\varepsilon)|^{-1}\Big).$$

In case F is an integer polynomial, this bound becomes

$$\widetilde{O}\Big(n^2(\tau_F + k + m) + nk\,\overline{\log}(\varepsilon^{-1})\Big).$$

The proof is found in Appendix C.

Theorem A gives a bit complexity bound in terms of $\widehat{S}$. We now investigate the natural ε-clusters in $\widehat{S}$. From the definition of $\widehat{S}$, we could write

$$\widehat{S} = S\cup S' \tag{13}$$

where S is the set of natural ε-clusters defined by the confined leaves of $\widehat{\mathcal{T}}_{comp}$, and S' is the set of all the natural ε-cluster $\phi(B)$ with B being any constituent box of any non-confined component in the preprocessing stage. Now we want to show an intrinsic property of the output components and also of the set $\widehat{S}$, using the concept of strong ε-clusters as is defined in the introduction.

Theorem B *Each natural ε-cluster in $\widehat{S}$ is a union of strong ε-clusters.*

The proof of Theorem B is found in Appendix C.

7. Conclusion

This paper initiates the investigation of the local complexity of root clustering. It modifies the basic analysis and techniques of Becker et al. (2017) to achieve this. Moreover, it solves a problem left open in Becker et al. (2017), which is to show that our complexity bounds can be achieved without adding a preprocessing step to search for "nice boxes" containing roots.

We mention some open problems. Our Theorem A expresses the complexity in terms of local geometric parameters; how tight is this? Another challenge is to extend our complexity analysis to analytic root clustering Yap et al. (2013).

References

Abbott, J., 2014. Quadratic interval refinement for real roots. ACM Communications in Computer Algebra 48 (1), 3–12.

Becker, R., May 2012. The Bolzano Method to isolate the real roots of a bitstream polynomial. Bachelor thesis, University of Saarland, Saarbruecken, Germany.

Becker, R., Sagraloff, M., Sharma, V., Yap, C., Sep. 2017. A near-optimal subdivision algorithm for complex root isolation based on Pellet test and Newton iteration. J. Symbolic ComputationHttps://doi.org/10.1016/j.jsc.2017.03.009. Also: arXiv:1509.06231v3 [cs.NA], 51 Pages.

Burr, M., Krahmer, F., 2012. SqFreeEVAL: An (almost) optimal real-root isolation algorithm. J. Symbolic Computation 47 (2), 153–166.

Burr, M., Krahmer, F., Yap, C., December 2009. Continuous amortization: A non-probabilistic adaptive analysis technique. Electronic Colloquium on Computational Complexity (ECCC) TR09 (136).
URL `http://eccc.hpi-web.de/report/2009/136/`

Burr, M. A., 2016. Continuous amortization and extensions: With applications to bisection-based root isolation. J. Symbolic Computation 77, 78–126.

Collins, G. E., Akritas, A. G., 1976. Polynomial real root isolation using Descartes' rule of signs. In: Jenks, R. D. (Ed.), Proceedings of the 1976 ACM Symposium on Symbolic and Algebraic Computation. ACM Press, pp. 272–275.

Davenport, J. H., 1985. Computer algebra for cylindrical algebraic decomposition. Tech. Rep., The Royal Inst. of Technology, Dept. of Numerical Analysis and Computing Science, S-100 44, Stockholm, Sweden, reprinted as Tech. Report 88-10 , School of Mathematical Sci., U. of Bath, Claverton Down, Bath BA2 7AY, England. URL http://www.bath.ac.uk/˜masjhd/TRITA.pdf.

Du, Z., Sharma, V., Yap, C., 2007. Amortized bounds for root isolation via Sturm sequences. In: Wang, D., Zhi, L. (Eds.), Symbolic-Numeric Computation. Trends in Mathematics. Birkhäuser Verlag AG, Basel, pp. 113–130, proc. Int'l Workshop on Symbolic-Numeric Computation, Xi'an, China, Jul 19–21, 2005.

Eigenwillig, A., Sharma, V., Yap, C., 2006. Almost tight complexity bounds for the Descartes method. In: 31st Int'l Symp. Symbolic and Alge. Comp. (ISSAC'06). pp. 71–78, genova, Italy. Jul 9-12, 2006.

Emiris, I. Z., Pan, V. Y., Tsigaridas, E. P., 2014. Algebraic algorithms. In: Gonzalez, T., Diaz-Herrera, J., Tucker, A. (Eds.), Computing Handbook, 3rd Edition: Computer Science and Software Engineering. Chapman and Hall/CRC, pp. 10: 1–30.

Kamath, N., Aug. 2010. Subdivision algorithms for complex root isolation: Empirical comparisons. Msc thesis, Oxford University, Oxford Computing Laboratory.

Kamath, N., Voiculescu, I., Yap, C., 2011. Empirical study of an evaluation-based subdivision algorithm for complex root isolation. In: 4th Intl. Workshop on Symbolic-Numeric Computation (SNC). pp. 155–164.

Kirrinnis, P., 1998. Polynomial factorization and partial fraction decomposition by simultaneous Newton's iteration. J. of Complexity 14, 378–444.
Lane, J. M., Riesenfeld, R. F., 1981. Bounds on a polynomial. BIT 21, 112–117.
Marden, M., 1949. The Geometry of Zeros of a Polynomial in a Complex Variable. Math. Surveys. American Math. Soc., New York.
McNamee, J., Pan, V., 2013. Numerical Methods for Roots of Polynomials, Part 2. Elsevier, Amsterdam.
Mehlhorn, K., Sagraloff, M., Wang, P., 2015. From approximate factorization to root isolation with application to cylindrical algebraic decomposition. J. Symbolic Computation 66, 34–69.
URL http://dx.doi.org/10.1016/j.jsc.2014.02.001
Neff, C. A., Reif, J. H., 1996. An efficient algorithm for the complex roots problem. J. of Complexity 12, 81–115.
Pan, V. Y., 1997. Solving a polynomial equation: some history and recent progress. SIAM Review 39 (2), 187–220.
Pan, V. Y., 2000. Approximating polynomial zeros: Modified quadtree (Weyl's) construction and improved Newton's iteration. J.Complexity 16 (1), 213–264.
Pan, V. Y., 2002. Univariate polynomials: Nearly optimal algorithms for numerical factorization and root-finding. J. Symb. Comput. 33 (5), 701–733.
Renegar, J., 1987. On the worst-case arithmetic complexity of approximating zeros of polynomials. Journal of Complexity 3, 90–113.
Rouillier, F., Zimmermann, P., 2004. Efficient isolation of [a] polynomial's real roots. J. Computational and Applied Mathematics 162, 33–50.
Sagraloff, M., 2012. When Newton meets Descartes: A simple and fast algorithm to isolate the real roots of a polynomial. In: 37th Int'l Symp. Symbolic and Alge. Comp. pp. 297–304.
Sagraloff, M., Mehlhorn, K., 2015. Computing real roots of real polynomials. J. Symbolic Computation.
Sagraloff, M., Yap, C. K., 2011. A simple but exact and efficient algorithm for complex root isolation. In: Emiris, I. Z. (Ed.), 36th Int'l Symp. Symbolic and Alge. Comp. pp. 353–360, june 8-11, San Jose, California.
Schönhage, A., 1982. The fundamental theorem of algebra in terms of computational complexity. Manuscript, Department of Mathematics, University of Tübingen. Updated in 2004 with typo corrections and an appendix of related subsequent papers.
URL www.informatik.uni-bonn.de/~schoe/fdthmrep.ps.gz
Sharma, V., Yap, C., 2012. Near optimal tree size bounds on a simple real root isolation algorithm. In: 37th Int'l Symp. Symbolic and Alge. Comp.(ISSAC'12). pp. 319 – 326, jul 22-25, 2012. Grenoble, France.
Yakoubsohn, J.-C., 2000. Finding a cluster of zeros of univariate polynomials. Journal of Complexity 16 (3), 603–638.
Yap, C., Sagraloff, M., Sharma, V., 2013. Analytic root clustering: A complete algorithm using soft zero tests. In: Bonizzoni, P., Brattka, V., Lowe, B. (Eds.), Computability in Europe (CiE2013). Vol. 7921 of Lect. Notes in C.S. Springer, Heidelberg, pp. 434–444, invited Talk. Special Session on "Computational Complexity in the Continuous World", July 1-5, Milan, Italy.
Yap, C. K., 2009. In praise of numerical computation. In: Albers, S., Alt, H., Näher, S. (Eds.), Efficient Algorithms. Vol. 5760 of Lect. Notes in C.S. Springer-Verlag, pp. 308–407.

We have omitted the three appendices which may be found in our full paper: Appendix A contain proofs for Section 2. Similarly, Appendix B and C are for Sections 5 and 6.

Appendix A. Root Bounds

To prove Theorem 2, we follow Becker et al. (2017) by proving three lemmas. We then use these bounds to convert the bound in our Theorem A into a bound in terms of algebraic parameters as in (3) in Section 1.1.

Appendix A.1. LEMMA A1

In the following, we will define $G(z)$ and $H(z)$ relative to any Δ as follows:

$$F(z) = G(z)H(z) \tag{A.1}$$

where $G(z) = \prod_{i=1}(z-z_i)$ such that $Z_F(\Delta) = \texttt{Zero}(G) = \{z_1, \ldots, z_k\}$ and $\texttt{Zero}(H) = \{z_{k+1}, \ldots, z_n\}$. Note that the leading coefficients of $F(z)$ and $H(z)$ are the same. By induction on i, we may verify

that

$$F^{(i)}(z) = \sum_{j=0}^{i} \binom{i}{j} G^{(i-j)}(z) H^{(j)}(z)$$

and

$$\frac{F^{(i)}(z)}{i!} = \sum_{J \in \binom{[n]}{n-i}} \prod_{j \in J} (z - z_j).$$

LEMMA A1 *Let* $\Delta = \Delta(m, r)$ *and* $\lambda = \lambda_0 := 4k(n-k)$.
If $\#(\Delta) = \#(\lambda \cdot \Delta) = k \geq 0$ *then for all* $z \in \Delta$

$$\left| \frac{F^{(k)}(z)}{k! H(z)} \right| > 0.$$

For $z = m$, *the lower bound can be improved to half.*
Proof. Using the notation (A.1), we see that

$$\frac{F^{(k)}(z)}{k! H(z)} = \sum_{J \in \binom{[n]}{n-k}} \frac{\prod_{j \in J} (z - z_j)}{\prod_{i=k+1}^{n} (z - z_i)}$$

First suppose $\lambda_0 = 0$, i.e., $k = 0$ or $k = n$. If $k = n$, then $H(z)$ is the constant polynomial a_0 where a_0 is the leading coefficient of $F(z)$, and clearly, $\frac{F^{(k)}(z)}{k! H(z)} = 1$. If $k = 0$, then $F(z) = H(z)$ and again $\frac{F^{(k)}(z)}{k! H(z)} = 1$. In either case the lemma is verified.

Hence we next assume $\lambda_0 > 0$. We partition any $J \in \binom{[n]}{n-k}$ into $J' := J \cap [k]$ and $J'' := J \setminus [k]$. Then $j' := |J'|$ ranges from 0 to $\min(k, n-k)$. Also, $j' = 0$ iff $J = \{k+1, \ldots, n\}$.

$$\begin{aligned}
\frac{F^{(k)}(z)}{k! H(z)} &= \sum_{J \in \binom{[n]}{n-k}} \frac{\prod_{j \in J} (z - z_j)}{\prod_{i=k+1}^{n} (z - z_i)} \\
&= \sum_{j'=0}^{\min(k,n-k)} \sum_{J' \in \binom{[k]}{j'}} \sum_{J'' \in \binom{[n]\setminus[k]}{n-k-j'}} \frac{\prod_{i' \in J'} (z - z_{i'}) \prod_{i'' \in J''} (z - z_{i''})}{\prod_{i=k+1}^{n} (z - z_i)} \\
&= 1 + \sum_{j=1}^{\min(k,n-k)} \sum_{J' \in \binom{[k]}{j}} \sum_{J'' \in \binom{[n]\setminus[k]}{n-k-j}} \frac{\prod_{i' \in J'} (z - z_{i'}) \prod_{i'' \in J''} (z - z_{i''})}{\prod_{i=k+1}^{n} (z - z_i)}
\end{aligned}$$

We next show that the absolute value of the summation on the RHS is at most $\frac{20}{21}$ which completes the proof. Since $z, z_{i'} \in \Delta$, and $z_{i''} \notin 4k(n-k)\Delta$ it follows that $|z - z_{i'}| \leq 2r$ and $|z - z_{i''}| \geq 3k(n-k)r$.

From these inequalities, we get

$$\begin{aligned}
&\sum_{j=1}^{\min(k,n-k)} \sum_{J'\in\binom{[k]}{j}} \sum_{J''\in\binom{[n]\setminus[k]}{n-k-j}} \frac{\prod_{i'\in J'}|z-z_{i'}|\prod_{i''\in J''}|z-z_{i''}|}{\prod_{i=k+1}^{n}|z-z_i|}\\
&\le \sum_{j=1}^{\min(k,n-k)} \binom{k}{j}\binom{n-k}{n-k-j}\Big(\frac{2r}{3k(n-k)r}\Big)^j\\
&\le \sum_{j=1}^{\min(k,n-k)} \frac{k^j}{j!}\binom{n-k}{j}\Big(\frac{2}{3k(n-k)}\Big)^j\\
&< \sum_{j=1}^{k} \frac{1}{j!}\Big(\frac{2}{3}\Big)^j\\
&= e^{2/3}-1 < \frac{20}{21}.
\end{aligned}$$

For $z = m$, the term is upper bounded by $e^{1/4}-1 < \frac{1}{2}$.

Q.E.D.

Since for all $z \in \Delta$, $F^{(k)}(z) \neq 0$, we get the following:

Corollary A1 *Let $\lambda = \lambda_0 := 4k(n-k)$. If $\#(\Delta) = \#(\lambda\Delta) = k \ge 0$ then $F^{(k)}$ has no zeros in Δ.*

Appendix A.2. Lemma A2

Lemma A2 *Let $\Delta = \Delta(m,r)$, $\lambda = 4k(n-k)$ and $c_1 = 7kK$.*
If $\#(\Delta) = \#(\lambda\Delta) = k$ then

$$\sum_{i<k} \frac{|F^{(i)}(m)|}{|F^{(k)}(m)|}\frac{k!}{i!}(c_1 r)^{i-k} < \frac{1}{2K}.$$

Proof. The result is trivial if $k = 0$. We may assume that $k \ge 1$. With the notation of (A.1), we may write

$$\frac{|G^{(i)}(m)|}{i!} \le \sum_{J\in\binom{[k]}{k-i}} \prod_{j\in J}|m-z_j| \le \binom{k}{i} r^{k-i},$$

since $z_j \in \Delta$. Similarly, we obtain

$$\left|\frac{H^{(i)}(m)}{i!H(m)}\right| \le \sum_{J\in\binom{[n]\setminus[k]}{i}} \prod_{j\in J}\frac{1}{|m-z_j|} \le \binom{n-k}{i}\frac{1}{(\lambda r)^i}.$$

From these two results, we derive that

$$\begin{aligned}
\left|\frac{G^{(i-j)}(m)H^{(j)}(m)}{(i-j)!\,j!H(m)}\right| &\le \binom{k}{i-j} r^{k-(i-j)} \cdot \binom{n-k}{j}\frac{1}{(\lambda r)^j}\\
&= \binom{k}{i-j}\binom{n-k}{j}\cdot\frac{r^{k-i}}{\lambda^j}.\\
\binom{i}{j}\left|\frac{G^{(i-j)}(m)H^{(j)}(m)}{i!H(m)}\right| &\le \binom{k}{i-j}\binom{n-k}{j}\frac{r^{k-i}}{\lambda^j}.
\end{aligned}$$

Thus we get

$$
\begin{aligned}
&\sum_{i=0}^{k-1} \frac{|F^{(i)}(m)|}{|F^{(k)}(m)|} \frac{k!}{i!} (c_1 r)^{i-k} \\
&\le \sum_{i=0}^{k-1} \sum_{j=0}^{i} \frac{\binom{i}{j} |G^{(i-j)}(m) H^{(j)}(m)|}{|F^{(k)}(m)|} \frac{k!}{i!} (c_1 r)^{i-k} \\
&\le \sum_{i=0}^{k-1} \sum_{j=0}^{i} \frac{|H(m)|}{|F^{(k)}(m)|} \binom{k}{i-j} \binom{n-k}{j} \frac{k! c_1^{i-k}}{\lambda^j} \\
&\le 2 \sum_{i=0}^{k-1} \sum_{j=0}^{i} \binom{k}{k-i+j} \cdot \binom{n-k}{j} \frac{c_1^{i-k}}{\lambda^j} \qquad \text{(by Lemma A1 for } z = m\text{)} \\
&\le 2 \sum_{i=0}^{k-1} \sum_{j=0}^{i} \frac{(k^j)(k^{k-i})}{(k-i+j)!} \cdot \frac{(n-k)^j}{j!} \frac{c_1^{i-k}}{(4k(n-k))^j} \\
&= 2 \sum_{i=0}^{k-1} \frac{k^{k-i} c_1^{i-k}}{(k-i)!} \sum_{j=0}^{i} \frac{1}{j! 4^j} \\
&< 2 \sum_{i=0}^{k-1} \frac{k^{k-i} c_1^{i-k}}{(k-i)!} e^{1/4} \\
&< 2e^{1/4} \sum_{j=1}^{k} \frac{(k/c_1)^j}{j!} \\
&< 2e^{1/4} (e^{1/7K} - 1) \\
&< 2e^{1/4} \frac{1}{7K-1} \\
&\le 2e^{1/4} \frac{1}{6K} < \frac{1}{2K}.
\end{aligned}
$$

Q.E.D.

Appendix A.3. Lemma A3

LEMMA A3 *Let* $\lambda_1 = 3K(n-k) \cdot \max\{1, 4k(n-k)\} = 3K(n-1) \cdot \max\{1, \lambda_0\}$.
If $\#(\Delta) = \#(\lambda_1 \cdot \Delta) = k \ge 0$ *then*

$$\sum_{i=k+1}^{n} \left| \frac{F^{(i)}(m) r^{i-k} k!}{F^{(k)}(m) i!} \right| < \frac{1}{2K}.$$

where $\Delta = \Delta(m, r)$.

Proof. First, assume $\lambda_0 = 4k(n-k) > 0$ (i.e., $0 < k < n$). Let $\texttt{Zero}(F^{(k)}) = \left\{z_1^{(k)}, \ldots, z_{n-k}^{(k)}\right\}$ be the roots of $F^{(k)}$. Since

$$\#(3K(n-k)\Delta) = \#(3K(n-k) \cdot \lambda_0 \Delta),$$

Corollary A1 implies that $F^{(k)}$ has no roots in $3K(n-k)\cdot\Delta$. Thus, $|m-z_j^{(k)}| \geq 3K(n-k)r$ and

$$\begin{aligned}\left|\frac{F^{(k+i)}(m)}{F^{(k)}(m)}\right| &\leq i! \sum_{J\in\binom{[n-k]}{i}} \prod_{j\in J} \frac{1}{|m-z_j^{(k)}|} \\ &\leq \frac{i!\binom{n-k}{i}}{(3K(n-k)r)^i} \\ &\leq \frac{(n-k)^i}{(3K(n-k)r)^i} \\ &\leq \frac{1}{(3Kr)^i}.\end{aligned}$$

It follows that

$$\begin{aligned}&\sum_{j=k+1}^{n} \left|\frac{F^{(j)}(m)r^{j-k}k!}{F^{(k)}(m)\,j!}\right| \\ &\leq \sum_{i=1}^{n-k} \left|\frac{F^{(k+i)}(m)}{F^{(k)}(m)}\right| \frac{r^i}{i!} \qquad \left(\text{ since } \frac{k!}{(k+i)!} \leq \frac{1}{i!}\right) \\ &\leq \sum_{i=1}^{n-k} \frac{1}{(3Kr)^i}\frac{r^i}{i!} \\ &\leq \sum_{i=1}^{n-k} \left(\frac{1}{3K}\right)^i \frac{1}{i!} \\ &< e^{1/3K} - 1 < \frac{1}{3K-1} < \frac{1}{2K}.\end{aligned}$$

It remains to consider the case $k=0$ or $k=n$. The lemma is trivial for $k=n$. When $k=0$, we have $\lambda_1 = 3Kn$ and the roots $z_j^{(k)}$ are the roots of F. Then $|m-z_j^{(k)}| \geq 3Knr$ follows from our assumption that $\#(\lambda_1\Delta) = \#(\Delta) = 0$. The preceding derivation remains valid. **Q.E.D.**

Corollary A3 *Let $c_1 \geq 1$. If $\#(\Delta) = \#(c_1\lambda_1\cdot\Delta) = k \geq 0$ then*

$$\sum_{i=k+1}^{n} \left|\frac{F^{(i)}(m)(c_1 r)^{i-k}k!}{F^{(k)}(m)i!}\right| < \frac{1}{2K}.$$

where $\Delta = \Delta(m,r)$.

Proof. Let $\Delta_1 = c_1\Delta$. Then $\#(\Delta_1) = \#(\lambda_1\Delta_1) = k$, and the previous lemma yields our conclusion (replacing r by $c_1 r$). **Q.E.D.**

Appendix A.4. Theorem 2

Theorem 2 *Let k be an integer with $0 \leq k \leq n = \deg(F)$ and $K \geq 1$.*
Let $c_1 = 7kK$, and $\lambda_1 = 3K(n-k)\cdot\max_1\{4k(n-k)\}$.

If

$$\#(\Delta) = \#(c_1\lambda_1\Delta) = k,$$

then

$$T_k(c_1\Delta, K, F) \quad \textit{holds.}$$

Proof.
By definition, $T_k(c_1\Delta, K, F)$ holds iff

$$\sum_{i\neq k} \frac{|F^{(i)}(m)(c_1 r)^{i-k}k!}{|F^{(k)}(m)|} < \frac{1}{K}$$

But the LHS is equal to $A + B$ where

$$A: \qquad \sum_{i>k} \frac{|F^{(i)}(m)(c_1 r)^{i-k}k!}{|F^{(k)}(m)|}$$

$$B: \qquad \sum_{i<k} \frac{|F^{(i)}(m)(c_1 r)^{i-k}k!}{|F^{(k)}(m)|}$$

By Corollary A3, A is at most $\frac{1}{2K}$ and by Lemma A2, B is at most $\frac{1}{2K}$. This proves our theorem. **Q.E.D.**

Appendix A.5. Bound on T_D in the Theorem A

We will need the following result to derive the bound.

LEMMA A4 *Let $g(x)$ be a complex polynomial of degree n with distinct roots $\alpha_1, \ldots, \alpha_m$ where α_i has multiplicity n_i. Thus $n = \sum_{i=1}^m n_i$. Let $I \subseteq [m]$ and $\nu = \min\{n_i : i \in I\}$. Then*

$$\prod_{i\in I} |g_{n_i}(\alpha_i)| \geq |\operatorname{GenDisc}(g)| \left(\|g\|_\infty^m n^{n+1} \operatorname{Mea}(g)^{n+1-\nu}\right)^{-1},$$

where

$$\operatorname{GenDisc}(g) := \operatorname{lcf}(g)^m \prod_{1\leq i<j\leq m} \left(\alpha_i - \alpha_j\right)^{n_i+n_j}$$

and $g_{n_i}(\alpha_i) := g^{(n_i)}(\alpha_i)/n_i!$.

Proof. From the observation that

$$g_{n_i}(\alpha_i) = \operatorname{lcf}(g) \prod_{1\leq j\leq m,\ j\neq i} (\alpha_i - \alpha_j)^{n_j},$$

we obtain the following relation:

$$\prod_{i=1}^m g_{n_i}(\alpha_i) = \operatorname{lcf}(g)^m \prod_{1\leq i<j\leq m} \left(\alpha_i - \alpha_j\right)^{n_i+n_j} = \operatorname{GenDisc}(g).$$

From this it follows that

$$\prod_{i\in I} |g_{n_i}(\alpha_i)| = |\operatorname{GenDisc}(g)| \left(\prod_{i\in[m]\setminus I} |g_{n_i}(\alpha_i)| \right)^{-1}. \tag{A.2}$$

We next derive an upper bound on $|g_{n_i}(\alpha_i)|$. Let $g(x) = \sum_{j=0}^{n} b_j x^j$. By standard arguments we know that

$$g_{n_i}(\alpha_i) = \sum_{j=n_i}^{n} \binom{j}{n_i} b_j \alpha_i^{j-n_i}.$$

Taking the absolute value and applying triangular inequality, we get

$$|g_{n_i}(\alpha_i)| \le \|g\|_\infty \sum_{j=n_i}^{n} \binom{j}{n_i} \max\{1, |\alpha_i|\}^{j-n_i}.$$

Applying Cauchy-Schwarz inequality to the RHS we obtain

$$|g_{n_i}(\alpha_i)| \le \|g\|_\infty \left(\sum_{j=n_i}^{n} \binom{j}{n_i}^2 \right)^{\frac{1}{2}} \left(\sum_{j=n_i}^{n} \max\{1, |\alpha_i|\}^{2(j-n_i)} \right)^{\frac{1}{2}}.$$

The second term in brackets on the RHS is smaller than $\max\{1, |\alpha_i|\}^{n-n_i+1}$, and the first is bounded by $\sum_{j=n_i}^{n} \binom{j}{n_i} = \binom{n+n_i+1}{n} \le n^{n_i+1}$. Thus we obtain

$$|g_{n_i}(\alpha_i)| \le \|g\|_\infty n^{n_i+1} \max\{1, |\alpha_i|\}^{n-n_i+1}.$$

Taking the product over all $i \in [m] \setminus I$, we get that

$$\prod_{i\in[k]\setminus I} |g_{n_i}(\alpha_i)| \le \|g\|_\infty^m n^{n+1} \mathrm{Mea(g)}^{\mathrm{n+1-min_{i\in I}\, n_i}}.$$

Substituting this upper bound in (A.2) yields us the desired bound. **Q.E.D.**

Let $I \subseteq [m]$. We next derive an upper bound on $\sum_{D\in\widehat{S}} T_D$, where

$$T_D = \overline{\log} \prod_{z_j\notin D} |\xi_i - z_j|^{-n_j},$$

here ξ_i is a representative root in the natural ε-cluster D. In this section, we use the convenient shorthand ξ_D to denote the representative for cluster D, and k_D the number of roots in D. Moreover, we choose the representative ξ_D as a root that has the smallest absolute value among all roots in D. Let $\mathcal{D}$ denote a set of disjoint *natural* ε-clusters of F such that the union of these clusters contains all the roots of F. Define F_ε as the polynomial obtained by replacing each natural ε-cluster D of F by its representative ξ_D with multiplicity k_D, i.e.,

$$F_\varepsilon(z) := \operatorname{lcf}(F) \prod_{D\in\mathcal{D}} (z - \xi_D)^{k_D}$$

More importantly, the choice of the representative ensures that the Mahler measure does not increase, i.e., $\mathrm{Mea(F_\varepsilon)} \le \mathrm{Mea(F)}$. Since ξ_D is a root of multiplicity k_D, it can be verified that

$$\frac{F_\varepsilon^{(k_D)}(\xi_D)}{k_D!} = \operatorname{lcf}(F) \prod_{D'\in\mathcal{D}, D'\neq D} (\xi_D - \xi_{D'})^{k_{D'}}.$$

We first relate the product $\prod_{z\notin D}|\xi_D - z_j|^{n_j}$ appearing in T_i with the term on the RHS above. The two are not the same, since we have replaced all natural ε-clusters with their representative, and hence for another cluster D' the distance $|\xi_D - z_j|$, for $z_j \in D'$, is not the same as $|\xi_D - \xi_{D'}|$. Nevertheless, for an isolator Δ' of D', we have

$$2\min_{w\in\Delta'}|\xi_D - w| \geq \max_{w\in\Delta'}|\xi_D - w|$$

and hence

$$|\xi_D - z_j| \geq \frac{|\xi_D - \xi_{D'}|}{2}.$$

From this inequality, we obtain that

$$\prod_{z_j\notin D}|\xi_D - z_j|^{n_j} \geq 2^{-n}\frac{\left|F_\varepsilon^{(k_D)}(\xi_D)\right|}{k_D!}.$$

So to derive an upper bound on $\sum_{D\in\widehat{S}} T_D$, it suffices to derive a lower bound on $\prod_{D\in\widehat{S}}|F_\varepsilon^{(k)}(\xi_D)|/k!$. Applying the bound in Lemma A4 above to F_ε, along with the observations that $\|F_\varepsilon\|_\infty \leq 2^n \mathrm{Mea}(\mathrm{F}_\varepsilon)$, and $\mathrm{Mea}(\mathrm{F}_\varepsilon) \leq \mathrm{Mea}(\mathrm{F})$, we get the following result:

Theorem A5

$$\sum_{D\in\widehat{S}} T_D = \widetilde{O}(\overline{\log}\,|\,\mathrm{GenDisc}(F_\varepsilon)|^{-1} + nm + n\,\overline{\log}\,\mathrm{Mea}(\mathrm{F})).$$

Note, however, that

$$|\,\mathrm{GenDisc}(F_\varepsilon)| > \frac{|\,\mathrm{GenDisc}(F)|}{\varepsilon^{\sum_{D\in\widehat{S}} k_D^2}}.$$

If we assume that $\varepsilon < 1$, i.e., $|\,\mathrm{GenDisc}(F_\varepsilon)|$ is larger than $|\,\mathrm{GenDisc}(F)|$, then the term $(\sum_{D\in\widehat{S}} k_D^2)\log\varepsilon < 0$ and so we can replace $|\,\mathrm{GenDisc}(F_\varepsilon)|^{-1}$ by $|\,\mathrm{GenDisc}(F)|^{-1}$ in Theorem A5 to obtain a larger bound. Moreover, if F is an integer polynomial, not necessarily square-free, from (Mehlhorn et al., 2015, p. 52) we know that $\overline{\log}\,|\,\mathrm{GenDisc}(F)|^{-1} = O(n\tau_F + n\log n)$ Hence we obtain the following bound (using Landau's inequality $\mathrm{Mea}(F) \leq \|F\|_2 \leq n2^{\tau_F}$):

Corollary A6 *Let $\{D_i; i \in I \subseteq [m]\}$ be any set of disjoint nature ε-clusters of an integer polynomial F with m distinct roots. Then*

$$\sum_{i\in I} T_{D_i} = \widetilde{O}(n\tau_F + nm).$$

Appendix B. Bound on Number of Boxes

Appendix B.1. Lemma 8

Lemma 8 *Denote $k = \#(2B_0)$.*
(a) If C is a component in the pre-processing stage, then $w_C \geq \frac{w(B_0)}{48k}$.
(b) Suppose $C_1 \to \cdots \to C_2$ is a non-special path with $W_{C_1} < \varepsilon$. Then it holds

$$\frac{w_{C_1}}{w_{C_2}} < 57k.$$

(c) Let C be a confined leaf in $\widehat{\mathcal{T}}_{comp}$ then

$$w_C > \frac{\varepsilon}{2}\Big(\frac{1}{114k}\Big)^k.$$

Proof. (a) By way of contradiction, assume $w_C < \frac{w(B_0)}{48k}$. Then the parent component C' satisfies $w_{C'} < \frac{w(B_0)}{24k}$ since C is obtained from C' in a Bisection Step. Then $W_{C'} \leq 3kw_{C'} < \frac{w(B_0)}{8}$. Thus $C' \cap B_0$ is empty or C' is confined. In either case, we would not bisect C' in the pre-processing stage, contradicting the existence of C.

(b) In this proof and in the proof of part (c) of this Lemma, we write w_i, R_i, N_i, etc, instead of $w_{C_i}, R_{C_i}, N_{C_i}$, etc. By way of contradiction, assume that $\frac{w_1}{w_2} \geq 57k$. Since $W_1 \leq \varepsilon$, from the algorithm, we know that each step in the path $C_1 \to \cdots \to C_2$ is a Bisection step. Thus there exists a component C' such that $3k \cdot w_2 < w_{C'} \leq 6k \cdot w_2$. The following argument shows that C' is a leaf of $\widehat{\mathcal{T}}_{comp}$. By Lemma 7(a), we have $W_2 \leq 3kw_2$, thus $W_2 < w_{C'}$. Thus the roots in C' are contained in a square of width less than $w_{C'}$. By Lemma 7(b), we conclude that C' is compact. To show that C' is a leaf, it remains to show that $4\Delta_{C'}$ has no intersection with other components. We have $4R_{C'} = 4 \cdot \frac{3}{4}W_{C'} \leq 9w_{C'}$. Meanwhile, since C' is compact, it is easy to see that the distance from the center of $\Delta_{C'}$ to C' is at most $\frac{1}{2}w_{C'}$. Thus the separation between C' and any point in $4\Delta_{C'}$ is less than $9w_{C'} + \frac{1}{2}w_{C'} = \frac{19}{2}w_{C'} \leq \frac{19}{2} \cdot 6k \cdot w_2 \leq \frac{19}{2} \cdot 6k \cdot \frac{w_1}{57k} = w_1$. By Property (C3) in Section 3, we know that C' is separated from other components by at least w_1, thus $4\Delta_{C'}$ has no intersection with other components. We can conclude that C' is a leaf of $\widehat{\mathcal{T}}_{comp}$. Contradiction.

(c) Let C_0 be the first component above C such that $w_0 < \varepsilon$. From the algorithm, we have $w_0 \geq \frac{\varepsilon}{2}$. Consider the path $P = C_0 \to \cdots \to C$. There exists a consecutive sequence of special components below C_0, denoted as $\{C_1, \ldots, C_t\}$ with $C_t = C$. Split P into a concatenation $P = P_0; P_1; \cdots; P_{t-1}$ of t subpaths where subpath $P_i = (C_i \to \cdots C_{i+1})$ for $i \in \{0, \ldots, t-1\}$. Let C_i' be the parent of C_i in $\widehat{\mathcal{T}}_{comp}$ for $i \in \{1, \ldots, t\}$. Consider the subpath of P_i where we drop the last special configuration: $(C_i \to \cdots \to C'_{i+1})$. By part (b) of this lemma, we have

$$\frac{w_{C_i}}{w_{C'_{i+1}}} < 57k$$

for $i \in \{0, \ldots, t-1\}$. The step $C'_{i+1} \to C_{i+1}$ is evidently a Bisection step and so

$$\frac{w_i}{w_{i+1}} < 114k.$$

Hence $\frac{w_0}{w_t} < (114k)^k$. It follows $w_C > \frac{\varepsilon}{2}(\frac{1}{114k})^k$. **Q.E.D.**

Appendix B.2. Lemma 9

Before proving Theorem 9, we state the following lemma, which is an adaptation of (Becker et al., 2017, Lemma 8), giving a sufficient condition for the success of the Newton step.

LEMMA B1. *Let C be a confined component with $W_C \geq \varepsilon$. Then Newton(C) succeeds provided that*

(i) $\#(\Delta_C) = \#((2^{20} \cdot n^2 \cdot N_C) \cdot \Delta_C)$.

(ii) $\mathrm{rad}(\mathcal{Z}(C)) \le (2^{20} \cdot n)^{-1} \cdot \frac{R_C}{N_C}$.

THEOREM 9 *The length of the non-special path (6) satisfies*

$$s = O(\log\log \frac{w_1}{w_s} + \log n).$$

Particularly,

$$s_{\max} = O\Big(\log n + \log\log \frac{w(B_0)}{\varepsilon}\Big).$$

Proof. From Lemma 8(a), we can see that the length of path in the preprocessing stage is bounded by $O(\log n)$. From Lemma 8(b), the length of non-special path is bounded by $O(\log n)$ if the width of components is smaller than ε. Hence it remains to bound the length of non-special path in the main loop such that any component C in the path satisfies $W_C \ge \varepsilon$. Lemma B2 gives us the sufficient conditions to perform Newton step in this path.

As in Becker et al. (2017), the basic idea is to divide the path $P = (C_1 \to \cdots \to C_s)$ (using the notation of (6)) into 2 subpaths $P_1 = (C_1 \to \cdots \to C_{i_1})$ and $P_2 = (C_{i_1} \to \cdots \to C_s)$ such that the performance of the Newton steps in P_2 can be controlled by Lemma B1. This lemma has two requirements ((i) and (ii)): we show that the components in P_2 automatically satisfies requirement (i). Thus if component C_i in P_2 satisfies requirement (ii), we know that $C_i \to C_{i+1}$ is a Newton step. This allows us to bound the length of P_2 using the Abbot-Sagraloff Lemma (Becker et al., 2017, Lemma 9).

We write w_i, R_i, N_i, etc, instead of $w_{C_i}, R_{C_i}, N_{C_i}$, etc.

Define i_1 as to be the first index satisfying $N_{i_1} \cdot w_{i_1} < 2^{-24} \cdot n^{-3} \cdot w_1$. If no such index exists, take i_1 as s.

First we show that the length of P_1 is $O(\log n)$. Note that $N_i \cdot w_i$ decreases by a factor of at least 2 in each step Becker et al. (2017). There are two cases: if step $C_i \to C_{i+1}$ is a Bisection step, $w_{i+2} = w_i/2$ and N_i does not increase; if it is a Newton step, then $w_{i+1} = \frac{w_i}{2N_i}$ and $N_{i+1} = N_i^2$, so $N_{i+1} \cdot w_{i+1} = N_i^2 \cdot \frac{w_i}{2N_i} = \frac{1}{2} \cdot N_i \cdot w_i$. It follows that at most $\log(2^{24} \cdot n^3)$ steps are performed to reach an i' such that $N_{i'} \cdot w_{i'} \le 2^{-24} \cdot n^{-3} \cdot N_1 \cdot w_1$. This proves $i' \le 1 + \log(2^{24} \cdot n^3)$. Since C_1 is a special component, our algorithm reset $N_1 = 4$ (cf. proof of Lemma 7). So it takes 2 further steps from i' to satisfy the condition of i_1. Thus $i_1 \le 3 + \log(2^{24} \cdot n^3) = O(\log n)$. Note that this bound holds automatically if $i_1 = s$.

We now show that requirement (i) of Lemma B2 is satisfied in P_2: from the definition of i_1, for any $i \ge i_1$, $2^{20} \cdot n^2 \cdot N_i \cdot r_i \le 2^{20} \cdot n^2 \cdot N_i \cdot \frac{3}{4} \cdot 9n \cdot w_i < w_1$, and the separation of C_1 from any other component is at least w_1, so $(2^{20} \cdot n^2 \cdot N_i) \cdot \Delta_i$ contains only the roots in $\mathcal{Z}(C_1)$, fulfilling requirement (i).

Next consider the path P_2. Each step either takes a bisection step or a Newton step. However, it is guaranteed to take the Newton step if requirement (ii) holds (note that it may take a Newton step even if requirement (ii) fails). Let $\#(\Delta_s) = k$. If component C_i satisfies

$$\frac{R_i}{N_i} \ge 2^{20} \cdot n \cdot R_s, \tag{B.1}$$

the requirement (ii) is satisfied. But $R_s < \frac{3}{4} \cdot 9n \cdot w_s < 2^4 \cdot n \cdot w_s$ and $R_i \geq w_i$ so if

$$\frac{w_i}{N_i} \geq (2^{20} \cdot n) \cdot (2^4 \cdot n) \cdot w_s = 2^{24} \cdot n^2 \cdot w_s \tag{B.2}$$

holds, it would imply (B.1). On the other hand, (B.2) is precisely the requirement that allows us to invoke (Becker et al., 2017, Lemma 9). Applying that lemma bounds the length of P_2 by $A := (\log\log N_{i_1} + 2\log\log(w_{i_1} \cdot (2^{24} \cdot n^2)^{-1} \cdot \frac{1}{w_{C_s}}) + 2) + (2\log n + 24)$. Since $N_{i_1} \leq \frac{w_{i_1}}{w_s}$, we conclude that $A = O(\log\log \frac{w_{i_1}}{w_s} + \log n)$. This concludes our proof.

The second part of this theorem is a direct result from the first part. **Q.E.D.**

Appendix B.3. Lemma 11

We first prove two lemmas that is useful for later proof.

LEMMA B2. *Let C_1 be the parent of C_2 in $\mathcal{T}^*_{comp}$, then*

$$r_{C_1} \leq 3\sqrt{2}n \cdot w_{C_2}$$

Proof. Suppose C_2' is the parent of C_2 in the component tree $\widehat{\mathcal{T}}_{comp}$. Then all the roots in C_1 remain in C_2', meaning that $r_{C_2'} = r_{C_1}$. It is easy to see that the step $C_2' \to C_2$ is a Bisection Step, thus $w_{C_2'} = 2w_{C_2}$. By Lemma 7(a), we have $W_{C_2'} \leq 3n \cdot w_{C_2'} = 6n \cdot w_{C_2}$. It follows $r_{C_2'} \leq \frac{1}{2} \cdot \sqrt{2} W_{C_2'} \leq 3\sqrt{2}n \cdot w_{C_2}$. Hence $r_{C_1} = r_{C_2'} \leq 3\sqrt{2}n \cdot w_{C_2}$. **Q.E.D.**

LEMMA B3.
(a) For any box B produced in the preprocessing stage, if $\phi_0(B)$ is a natural ε-cluster, then we have $w_B \geq 2 \cdot \mathrm{rad}(\phi_0(B))$. (b) For any $B \neq \frac{5}{4}B_0$ produced in the algorithm, $\phi_0(B) \subseteq 2B_0$.

Proof. (a)

$$\begin{array}{rcll} w_B & \geq & \frac{w(B_0)}{48n} & \text{(by Lemma 8(a))} \\ & \geq & 2 \cdot \varepsilon & \text{(by small } \varepsilon \text{ assumption)} \\ & \geq & 2 \cdot \mathrm{rad}(\phi_0(B)) & \text{(by definition of } \varepsilon\text{-cluster)} \end{array}$$

(b) If $\phi_0(B)$ is a special component, it is easy to see that $\phi_0(B) \subseteq 2B_0$.

We now discuss the case where $\phi_0(B)$ is a natural ε-cluster. To show that $\phi_0(B) \subseteq 2B_0$, note that since B is a proper subbox of $\frac{5}{4}B_0$, it follows that $2B \subseteq \frac{15}{8}B_0$. Thus there is a gap of $\frac{w(B_0)}{16}$ between the boundaries of $2B_0$ and $\frac{15}{8}B_0$. Since $\phi_0(B)$ is a ε-cluster, thus $\mathrm{rad}(\phi_0(B)) < \varepsilon \leq \frac{w(B_0)}{96n}$, and $\phi_0(B) \cap 2B$ is non-empty, we conclude that $\phi_0(B)$ is properly contained in $2B_0$. **Q.E.D.**

LEMMA 11 *The total number of boxes in all the components in $\widehat{\mathcal{T}}_{comp}$ is*

$$O(t \cdot s_{\max}) = O(\#(2B_0) \cdot s_{\max})$$

with $t = |\{\phi_0(B) : B \text{ is any box in } \widehat{\mathcal{T}}_{comp}\}|$.

Proof. By the discussion above, we charge each box B to $\phi_0(B)$ which can be a special component or a cluster.

First consider the case where $\phi_0(B)$ is special component. Note that $\frac{1}{3}r_{\phi_0(B)} < w_B$. We claim that the number of boxes congruent with B that are charged to $\phi_0(B)$ is at most 64: to see this,

note that $2B \cap \mathcal{Z}(\phi_0(B))$. If Δ is the minimum disc containing $\mathcal{Z}(\phi_0(B))$, then $2B$ must intersect Δ. By some simple calculations, we see that at most 64 aligned boxes congruent to B can be charged to $\phi_0(B)$.

We now analyze the number of different sizes of the boxes that are charged to the same special component C.

Denote the parent of C in the special component tree $\mathcal{T}^*_{comp}$ as C'. Let B be a box such that $\phi_0(B) = C$ and suppose B is the constituent boxes of the component C_B, evidently, $w_B = w_{C_B}$. From the definition of ϕ_0, B satisfies one of the two following conditions: (i) C_B is an component in the path $C' \to \cdots \to C$ and $w_B > \frac{1}{3} r_C$; (ii) C_B is a component above C' and $\frac{1}{3} r_{C'} \geq w_B > \frac{1}{3} r_C$. It is easy to see that there number of components C_B satisfying condition (i) is bounded by $s_{\max}$ from Theorem 9. It remains to count the number of components C_B that satisfy condition(ii). By Lemma B2, we have $r_{C'} \leq 3\sqrt{2}n \cdot w_C$. Since B is charged to C but not C', we have $w_B \leq \frac{1}{3} \cdot r_{C'} \leq \sqrt{2}n \cdot w_C$. The box B is constitute an ancestor of C, thus $w_C \leq w_B$. Therefore, we have $w_C \leq w_B \leq \sqrt{2}n \cdot w_C$, and note that w_B decreases by a factor of at least 2 at each step, so w_B may take $\log(\sqrt{2}n)$ different values. Hence, the number of boxes charged to each special component is bounded by $64 s_{\max}$.

Now consider the case where a box is charged to a natural ε-cluster, this case only happens in preprocessing step where the number of steps is bounded by $O(\log n)$. On the other hand, by Lemma B3(a), we have $2\mathrm{rad}(\phi_0(B)) \leq w_B$ if $\phi_0(B)$ is a ε-cluster. Thus the number of boxes of the same size charged to a natural ε-cluster by ϕ_0 is at most 9. Therefore, the number of boxes charged to a natural ε-cluster by ϕ_0 is bounded by $O(\log n)$.

Thus we can conclude that the total number of boxes is bounded by $O(t \cdot s_{\max})$ with $t = |\{\phi_0(B) : B \text{ is any box in } \widehat{\mathcal{T}}_{comp}\}|$. **Q.E.D.**

Appendix C. Bit Complexity

Appendix C.1. Lemma 14

We first prove two lemmas for later use.

LEMMA C1. *Let $\Delta = \Delta(m, R)$ and $\widehat{\Delta} := K\Delta$ for some $K \geq 1$. Let D be any subset of $\mathcal{Z}(\widehat{\Delta})$ and $\zeta \in D$. If $\widehat{\mu} = \#(\widehat{\Delta})$ and $k_D = \#(D)$ then*

$$\max_{z \in \Delta} |F(z)| > R^{k_D} \cdot n^{-\widehat{\mu}} \cdot K^{-\widehat{\mu}+k_D} \cdot 2^{-3n+1} \cdot \prod_{z_j \notin D} |\zeta - z_j|^{n_j}.$$

where z_j ranges over all the roots of F outside D and $\#(z_j) = n_j$.

Proof. Let $\{z_1, z_2, \ldots, z_r\}$ be the set of all the distinct roots of F. Wlog, assume that ζ appearing in the lemma is z_1. There exists a point $p \in \Delta(m, \frac{R}{2})$ such that the distance from p to any root of F is at least $\frac{R}{2n}$, this is because the union of all discs $\Delta(z_i, \frac{R}{2n})$ covers an area of at most $n \cdot \pi(\frac{R}{2n})^2 = \pi\frac{R^2}{4n} < \pi(\frac{R}{2})^2$. Then for a root $z_i \in \widehat{\Delta}$, it holds $\frac{|p-z_i|}{|z_1-z_i|} \geq \frac{R/(2n)}{2KR} = \frac{1}{4nK}$, and for a root $z_j \notin \widehat{\Delta}$, it holds $\frac{|p-z_j|}{|z_1-z_j|} \geq \frac{|p-z_j|}{|p-z_j|+|p-z_1|} = \frac{1}{1+\frac{|p-z_1|}{|p-z_j|}} \geq \frac{1}{1+\frac{2KR}{KR-R/2}} = \frac{1}{5}$. Note that

$|F(p)| = \mathrm{lcf}(F) \cdot \prod_{i=1}^{r} |p - z_i|^{n_i}$, it follows

$$\begin{aligned}
&\frac{|F(p)|}{\prod_{z_j \notin D} |z_1 - z_j|^{n_j}} \\
&= \mathrm{lcf}(F) \prod_{z_i \in D} |p - z_i|^{n_i} \prod_{z_j \in \widehat{\Delta}, z_j \notin D} \left| \frac{p - z_j}{z_1 - z_j} \right|^{n_j} \prod_{z_k \notin \widehat{\Delta}} \left| \frac{p - z_k}{z_1 - z_k} \right|^{n_k} \\
&\geq \frac{1}{4} \cdot \left(\frac{R}{2n} \right)^{k_D} \cdot \left(\frac{1}{4nK} \right)^{\widehat{\mu} - k_D} \cdot \left(\frac{1}{5} \right)^{n - \widehat{\mu}} \\
&> R^{k_D} \cdot n^{-\widehat{\mu}} \cdot K^{-\widehat{\mu} + k_D} \cdot 2^{-3n-1},
\end{aligned}$$

which proves the Lemma. **Q.E.D.**

LEMMA C2. *For any box B, $\phi(B)$ is contained in $14B$.*

Proof. Consider $\phi_0(B)$. If $\phi_0(B)$ is a cluster, then $2B$ intersects $\phi_0(B)$, and $2\mathrm{rad}(\phi_0(B)) \leq w_B$ (Lemma B3(a)). Thus $\phi_0(B) \subseteq 4B$.

Next suppose $\phi_0(B)$ is a special component. Then $w_B > \frac{1}{3} r_C$ where $r_C = \mathrm{rad}(\mathcal{Z}(C))$. Since $2B \cap \mathcal{Z}(C)$ is non-empty, we conclude that $\mathcal{Z}(C) \subseteq 14B$. **Q.E.D.**

Now we derive a bound for the cost of processing each component and box.

LEMMA 14 *Denote $k = \#(2B_0)$.*
(a) Let B be a box produced in the algorithm. The cost of processing B is bounded by

$$\widetilde{O}\Big(n \cdot [\tau_F + n \, \overline{\log}(B) + k_D \cdot (\overline{\log}(\varepsilon^{-1}) + k) + T_D] \Big) \tag{C.1}$$

with $D = \phi(B)$, $k_D = \#(D)$ and

$$T_D := \overline{\log} \prod_{z_j \notin D} |\xi_D - z_j|^{-n_j}. \tag{C.2}$$

where ξ_D is an arbitrary root contained in D.
(b) Let C be a component produced in the main-loop, and let C_0 be the last special component above C, then the cost of processing a component C is bounded by

$$\begin{aligned}
\widetilde{O}\Big(& n \cdot [\tau_F + n \, \overline{\log}(C) + n \, \overline{\log}(w_{C_0}) \\
& + k_D \cdot (\overline{\log}(\varepsilon^{-1}) + k) + T_D] \Big)
\end{aligned} \tag{C.3}$$

where D is an arbitrary cluster contained in C, $k_D = \#(D)$ and T_D is as defined in (C.2).

Proof. (a) According to (Becker et al., 2017, Lemma 7): the cost for carrying out a $\widetilde{T}^G(\Delta)$ test (associated with a box B or component C) is bounded by

$$\widetilde{O}\Big(n \cdot [\tau_F + n \cdot \overline{\log}(m, r) + L(\Delta, F)] \Big). \tag{C.4}$$

Thus for each call of $\widetilde{T}^G(\Delta)$ test, we need to bound $\overline{\log}(m, r)$ and $L(\Delta, F)$.

For $\widetilde{T}_0^G(\Delta(B))$, we need to perform $\widetilde{T}_0^G$ test for each subbox B_i into which B is divided. We have $\Delta_{B_i} = \Delta(m, r)$, it is easy to see that $\overline{\log}(m, r) \leq \overline{\log}(B)$. So it remains to bound the term

$L(\Delta, F)$ in (C.4). By definition, $L(\Delta, F) = 2 \cdot (4 + \overline{\log}(\|F_\Delta\|_\infty^{-1}))$ And for any $z \in \Delta$, it holds $|F(z)| \leq n \cdot \|F_\Delta\|_\infty$. Hence, we need to prove that $\overline{\log}((\max_{z\in\Delta_{B_i}} |F(z)|)^{-1})$ can be bounded by (C.1).

We apply Lemma C1 to obtain the bound of $\log((\max_{z\in\Delta_{B_i}}$ $|F(z)|)^{-1})$. Since $\phi(B) \subseteq \mathcal{Z}(14B \cap 2B_0)$ (Lemma C2), it suffices to take $\widehat{\Delta} = 42 \cdot \Delta_{B_i}$ since $42\Delta_{B_i}$ contains $14 \cdot \Delta_B$ which (by Lemma C2) contains $\phi(B)$. Hence with $K' = 42$, Lemma C1 yields that $\max_{z\in\Delta_B} |F(z)| > (\frac{3}{4} \cdot \frac{w_B}{2})^{k_D} \cdot n^{-\#(\widehat{\Delta})} \cdot (K')^{-\#(\widehat{\Delta})+k_D} \cdot 2^{-3n-1} \prod_{z_j\notin D} |\xi_D - z_j|^{n_j}$ where $D = \phi(B)$, $k_D = \#(D)$, and ξ_D is an arbitrary root contained in D. From Lemma 8(c), we have $w_B > \frac{\varepsilon}{2}(\frac{1}{114k})^k$. It is easy to check that $\overline{\log}((\max_{z\in\Delta_B} |F(z)|)^{-1})$ is bounded by (C.1).

(b) To bound the cost of processing a component C, we need to bound the cost of performing $\widetilde{T}^G(\Delta_C)$ and $\widetilde{T}^G(\Delta')$. It is easy to see that in both cases where $\Delta(m, r) = \Delta_C$ and $\Delta(m, r) = \Delta'$, we have $\overline{\log}(m, r) = O(\overline{\log}(C))$. With the same arguments in the proof of (a), it remains to prove that both $\log \max_{z\in\Delta_C |F(z)|^{-1}}$ and $\log \max_{z\in\Delta' |F(z)|^{-1}}$ are bounded by (C.3).

First consider the $\widetilde{T}^G_*(\Delta_C)$ test, by applying Lemma C1 with $K = 1$, we have $\max_{z\in\Delta} |F(z)| > R_C^{k_D} \cdot n^{-k_C} \cdot 2^{-3n-1} \cdot \prod_{z_j\notin D} |\xi_D - z_j|^{n_j}$ with D an arbitrary cluster in C, $k_D = \#(D)$ and ξ_D an arbitrary root in D. We know that $R_C \geq \frac{4}{3} w_C$. With the same arguments as in part (a), we can conclude that the cost of $\widetilde{T}^G_*(\Delta_C)$ test is bounded by (C.3).

Now consider $\widetilde{T}^G_{k_C}(\Delta')$ test with $\Delta' = \Delta(m', \frac{w_C}{8N_C})$ and m' as defined in the algorithm of Newton test. Here we take $\widehat{\Delta} = 2 \cdot 3n \cdot 8N_C \cdot \Delta' = 48nN_C \cdot \Delta'$ since $48nN_C\Delta'$ will contain C and thus contain all the roots in C. By applying Lemma C1 with $K = 48nN_C$, we have $\max_{\Delta'} |F(z)| > (\frac{w_C}{8N_C})^{k_D} \cdot n^{-\#(\widehat{\Delta})} \cdot K^{-\#(\widehat{\Delta})+k_D} \cdot 2^{-3n-1} \cdot \prod_{z_j\notin D} |\xi_D - z_j|^{n_j}$ with D an arbitrary cluster in C, $k_D = \#(D)$ and ξ_D an arbitrary root in D. First consider the lower bound for $(\frac{w_C}{8N_C})^{k_D}$. By lemma B0(b), we have $N_C \leq \frac{4w_{C_0}}{w_C}$, thus $\frac{w_C}{8N_C} \geq \frac{w_C^2}{32w_{C_0}}$. It follows $\overline{\log}(((\frac{w_C}{8N_C})^{k_D})^{-1}) = k_D(2\,\overline{\log}(w_C^{-1}) + \overline{\log}(w_{C_0}) + 5)$. As is proved, $k_D(2\,\overline{\log}(w_C) + \overline{\log}(w_{C_0}) + 5)$ is bounded by (C.3).

The bound for the other terms except $K^{\#(\widehat{\Delta})-k_D}$ are similar to the case discussed above. Hence it remains to bound $K^{\#(\widehat{\Delta})-k_D}$. Denote the radius of $\widehat{\Delta}$ as $\widehat{R}$, then $\widehat{R} = 18nw_C$ from the definition of $\widehat{\Delta}$. Note that $K = 48nN_C \leq 48n \cdot \frac{w_{C_0}}{w_C} = 48n \cdot 18n \cdot \frac{w_{C_0}}{\widehat{R}}$ and $\overline{\log}\left((48n \cdot 18n \cdot w_{C_0})^{\#(\widehat{\Delta})-k_D}\right) = O(n \log n + n\,\overline{\log}(w_{C_0}))$, thus it suffices to bound $\widehat{R}^{-\#(\widehat{\Delta})+k_D}$. For any root ξ_D of F in any ε-cluster $D \subseteq C$ which contains k_D roots counted with multiplicities, we have

$$\begin{aligned}
\prod_{z_i\notin D} |\xi_D - z_i|^{n_i} &= \prod_{z_j\in\widehat{\Delta}, z_j\notin D} |\xi_D - z_j|^{n_j} \prod_{z_k\notin\widehat{\Delta}} |\xi_D - z_k|^{n_k} \\
&\leq (2\widehat{R})^{\#(\widehat{\Delta})-k_D} \cdot \frac{\mathrm{Mea}(F(\xi_D + z))}{|\operatorname{lcf}(F)|} \\
&\leq (2(\widehat{R})^{\#(\widehat{\Delta})-k_D} \cdot 2^{\tau_F} 2^{n+3} \max_1(\xi_D)^n \\
&\leq 2^{\tau_F+2n+3} \cdot \max_1(\xi_D)^n \cdot \widehat{R}^{\#(\widehat{\Delta})-k_D}
\end{aligned}$$

So $\log(\widehat{R}^{-\#(\widehat{\Delta})+\xi_D})$ is bounded by (C.3). Hence the cost for processing component C, that is the two kind of $\widetilde{T}^G$ tests discussed above can be bounded by (C.3). **Q.E.D.**

Appendix C.2. Corollary to Theorem A

COROLLARY TO THEOREM A
The bit complexity of the algorithm is bounded by

$$\widetilde{O}\Big(n^2(\tau_F + k + m) + nk\,\overline{\log}(\varepsilon^{-1}) + n\,\overline{\log}\,|\,\mathrm{GenDisc}(F_\varepsilon)|^{-1}\Big).$$

In case F is an integer polynomial, this bound becomes

$$\widetilde{O}\Big(n^2(\tau_F + k + m) + nk\,\overline{\log}(\varepsilon^{-1})\Big).$$

Proof. From our assumption in Section 6, $\overline{\log}(B_0) = O(\tau_F)$. We can also see that $\sum_{D\in\widehat{S}} L_D \leq n\tau_F + k(k + \overline{\log}(\varepsilon^{-1})) + \sum_{D\in\widehat{S}} T_D + \sum_{i=1}^k \overline{\log}(z_i)$.

By Theorem A5, $\sum_{D\in\widehat{S}} T_D = \widetilde{O}(\overline{\log}\,|\,\mathrm{GenDisc}(F_\varepsilon)|^{-1} + nm + n\,\overline{\log}\,\mathrm{Mea(F)})$. And $\sum_{D\in\widehat{S}} \overline{\log}(\xi_D) \leq \sum_{i=1}^k \overline{\log}(z_i) \leq \overline{\log}\,\mathrm{Mea(F)} + k = O(\tau + k + \log n)$ (using Landau's inequality). From the equations above, we can deduce the first part of this lemma.

The second part comes from Corollary A6. **Q.E.D.**

Appendix C.3. Theorem B

We first show two useful lemmas: Lemma C3 is about root separation in components, and Lemma C4 says that strong ε-clusters are actually natural clusters.

LEMMA C3. *If C is any confined component, and its multiset of roots $\mathcal{Z}(C)$ is partitioned into two subsets G, H. Then there exists $z_g \in G$ and $z_h \in H$ such that $|z_g - z_h| \leq (2 + \sqrt{2})w_C$.*

Proof. We can define the $\mathcal{S}_G := \{B \in \mathcal{S}_C : 2B \cap G \neq \emptyset\}$ and $\mathcal{S}_H := \{B \in \mathcal{S}_C : 2B \cap H \neq \emptyset\}$. Note that $\mathcal{S}_G \cup \mathcal{S}_H = \mathcal{S}_C$. Since the union of the supports of $\mathcal{S}_G$ and $\mathcal{S}_H$ is connected, there must a box $B_g \in \mathcal{S}_G$ and $B_h \in \mathcal{S}_H$ such that $B_g \cap B_h$ is non-empty. This means that the centers of B_g and B_h are at most $\sqrt{2}w_C$ apart. From Corollary 5, there is root z_g (resp., z_h) at distance $\leq w_C$ from the centers of B_g (resp., B_h). Hence $|z_g - z_h| \leq (2 + \sqrt{2})w_C$. **Q.E.D.**

LEMMA C4. *Each strong ε-cluster is a natural ε-cluster.*

Proof. In the definition of ε-equivalence, if $z \overset{\varepsilon}{\sim} z'$ then there is a witness isolator Δ containing z and z'. If $z' \overset{\varepsilon}{\sim} z''$ we have another witness Δ' containing z' and z''. It follows from basic properties of isolators that if Δ and Δ' intersect, then there is inclusion relation between $\mathcal{Z}(\Delta)$ and $\mathcal{Z}(\Delta')$. Thus Δ or Δ' is a witness for $z \overset{\varepsilon}{\sim} z''$. Proceeding in this way, we eventually get a witness isolator for the entire equivalence class. **Q.E.D.**

Theorem B
Each natural ε-cluster in $\widehat{S}$ is a union of strong ε-clusters.

Proof. First we make an observation: For any strong ε-cluster D' and confined component C', if $D' \cap \mathcal{Z}(C') \neq \emptyset$ and $w_{C'} > 2 \cdot \mathrm{rad}(D')$, then $D' \subset \mathcal{Z}(C')$. To see this: suppose, $z_1 \in D' \cap \mathcal{Z}(C')$ and $z_2 \in \mathcal{Z}(D)$ belong to a component other than C'. By Property (C3), $|z_1 - z_2| \geq w_{C'} > 2r$, contradicting the fact that any 2 roots in D' are separated by distance at most $2r$.

Let $D \in \widehat{S}$. There are two cases: D is either in S or in S' where $\widehat{S} = S \cup S'$ as defined in (13).

First, assume that $D \in S'$. This case is relatively easy. Suppose E is a strong ε-cluster and $D \cap E \neq \emptyset$. From Lemma C4, E is also a natural cluster; thus either $D \subset E$ or $E \subset D$. By the definition of $\phi_0(B)$, D is a largest natural ε-cluster, meaning that there is no natural ε-cluster strictly containing D. Hence it follows $E \subset D$, which is what we wanted to prove.

In the remainder of this proof, we show that each natural ε-cluster in D is S is a union of strong ε-cluster. The observation above and Lemma B3(a) imply that for each component C' in the preprocessing stage, C' is a union of strong ε-clusters. Thus, when the mains loop starts, for each component C in Q_1, $\mathcal{Z}(C)$ is a union of strong ε-clusters.

Suppose D is a strong ε-cluster and C is a confined leaf of $\widehat{\mathcal{T}}_{comp}$. It is sufficient to prove that if $D \cap \mathcal{Z}(C) \neq \emptyset$, then $D \subseteq \mathcal{Z}(C)$. Let $r = \text{rad}(D)$. Suppose $z_1 \in D \cap \mathcal{Z}(C)$. There is an unique maximal path in $\widehat{\mathcal{T}}_{comp}$ such that all the components in this path contain z_1.

Consider the first component C_1 in the path above such that C_1 contains the root z_1 and $w_{C_1} \leq 4r$. If C_1 does not exist, it means that the leaf C_t in this path satisfies $w_{C_t} \geq 4r$, and by the observation above, it follows that $D \subseteq \mathcal{Z}(C_t)$. Henceforth assume C_1 exists; we will prove that it is actually a leaf of $\widehat{\mathcal{T}}_{comp}$.

Consider C_1', the parent of C_1 in $\widehat{\mathcal{T}}_{comp}$. Note that $w_{C_1'} \geq 4r$, and by the observation above, $D \subseteq \mathcal{Z}(C_1')$. We show that $w_{C_1} > 2r$. To show this, we discuss two cases. If the step $C_1' \to C_1$ is a Newton Step, then all the roots in C_1 are contained in a disc of radius $r' = \frac{w_{C_1'}}{8N_{C_1'}}$. Note that $r' \geq r$ since the Newton disc contains all the roots in C_1' and hence contains D. Newton step gives us $w_{C_1} = \frac{w_{C_1'}}{2N_{C_1'}} = 4r' \geq 4r$. If $C_1' \to C_1$ is a Bisection Step, then $w_{C_1} = w_{C_1'}/2 > 2r$. To summarize, we now know that $2r < w_{C_1} \leq 4r$. Again, from our above observation, we conclude that $D \subseteq \mathcal{Z}(C_1)$.

First a notation: let Δ_D be the smallest disc containing D. We now prove that $\mathcal{Z}(C_1) \subseteq D$. By way of contradiction, suppose there is a root $z \in \mathcal{Z}(C_1) \setminus D$. Since D is a strong ε-cluster, $\#(\Delta_D) = \#(114\Delta_D)$. It follows that for any $z' \in D$, we must have have $|z - z'| > 113r$. On the other hand, by Lemma C3, there exists z and z' fulfilling the above assumptions with the property that $|z - z'| \leq (2 + \sqrt{2})w_{C_1} \leq (2 + \sqrt{2})4r < 113r$. Thus we arrived at a contradiction.

From the above discussion, we conclude that $\mathcal{Z}(C_1) = D$ and $2r < w_{C_1} \leq 4r$, it is easy to see that $W_{C_1} \leq 3w_{C_1}$. Hence we can conclude that $W_{C_1} \leq 12r < 12 \cdot \frac{\varepsilon}{12} \leq \varepsilon$. Therefore, to show that C_1 is a leaf, it remains to prove that $4\Delta_{C_1} \cap C_2 = \emptyset$ for all C_2 in $Q_1 \cup Q_{dis}$.

Since $2r < w_{C_1} \leq 4r$, by some simple calculations, we can obtain that $C_1 \subset 8\Delta_D$ thus Δ_{C_1} is contained in $9\Delta_D$, it follows $4\Delta_{C_1} \subset 36\Delta_D$. It suffices to prove that $36\Delta_D \cap C_2 = \emptyset$ for all C_2. Note that for any root $z_1 \in C_1$ and any component C_2, we have $\text{Sep}(z_1, C_2) \geq w_{C_2}$ by property (C3). Assume that $\text{Sep}(z_1, C_2) = |z_1 - p|$ for some $p \in C_2$. We claim that there exists a root $z_2 \in C_2$ such that $|z_2 - p| \leq \frac{3\sqrt{2}}{2} w_{C_2}$. [To see this, suppose that p is contained in a constituent box B_2 of C_2, note that $2B_2$ must contain a root, assume that $z_2 \in 2B_2$, it follows $|z_2 - p| \leq \frac{3\sqrt{2}}{2} w_{C_2}$.] Hence $|z_1 - p| + |z_2 - p| \leq \text{Sep}(z_1, C_2) + \frac{3\sqrt{2}}{2} \cdot \text{Sep}(z_1, C_2)$. Note that $\#(\Delta_D) = \#(114\Delta_D)$, thus $|z_1 - z_2| \leq 113r$. By triangular inequality, we have $|z_1 - z_2| \leq |z_1 - p| + |z_2 - p| < (1 + \frac{3\sqrt{2}}{2}) \cdot \text{Sep}(z_1, C_2)$. Hence $\text{Sep}(z_1, C_2) \geq \frac{1}{1+3\sqrt{2}/2} |z_1 - z_2| > 36r$, implying $36\Delta_D \cap C_2 = \emptyset$.

This proves that our algorithm will output C_1, i.e., C_1 is a confined leaf of $\widehat{\mathcal{T}}_{comp}$.

In summary, each natural ε-cluster in $\widehat{S}$ is a union of strong ε-cluster. **Q.E.D.**

Appendix C.4. A complete proof of Theorem A

Based on Lemma 14, we can now derive the total cost of carrying out all the $\widetilde{T}^G$ tests in the algorithm.

A direct result from Lemma 14 is that the cost of processing all the boxes can be bounded by

$$\widetilde{O}\Big(\sum_{B\in\mathcal{B}}\Big(n\cdot[\tau_F+n\,\overline{\log}(B)+k_{\phi(B)}\cdot(\overline{\log}(\varepsilon^{-1})+k)+T_{\phi(B)}]\Big)\Big)$$

where $\mathcal{B}$ is the set of all the boxes produced in the algorithm.

Taking into account the fact that the number of boxes charged to a natural ϵ-cluster by the map ϕ is bounded by $O(s_{\max}\log n)=\widetilde{O}(1)$, we can write the above bound as

$$\widetilde{O}\Big(\sum_{B\in\mathcal{B}}n^2\,\overline{\log}(B)+\sum_{D\in\widehat{S}}\Big(n\cdot[\tau_F+k_D\cdot(\overline{\log}(\varepsilon^{-1})+k)+T_D]\Big)\Big). \tag{C.5}$$

Analogously, we can obtain that the cost of processing all the components can be bounded by

$$\widetilde{O}\Big(\sum_{C\in C}n^2\,(\overline{\log}(C)+\overline{\log}(w_{C_0}))+\sum_{D\in\widehat{S}}\Big(n\cdot[\tau_F+k_D\cdot(\overline{\log}(\varepsilon^{-1})+k)+T_D]\Big)\Big) \tag{C.6}$$

where C is the set of all the components produced in the algorithm and C_0 is the last special component above C.

The bounds (C.5) and (C.6) add up to the cost of processing all the boxes and components produced in the algorithm. To prove Theorem A, we want to show that both

$$\widetilde{O}\Big(\sum_{B\in\mathcal{B}}n^2\,\overline{\log}(B)\Big) \tag{C.7}$$

and

$$\widetilde{O}\Big(\sum_{C\in C}n^2\,(\overline{\log}(C)+\overline{\log}(w_{C_0}))\Big) \tag{C.8}$$

can be bounded by

$$\widetilde{O}\Big(n^2\,\overline{\log}(B_0)+n^2\sum_{D\in\widehat{S}}\overline{\log}(\xi_D)\Big) \tag{C.9}$$

where ξ_D is an arbitrary root contained in D.

First we show that the bound (C.8) can be bounded by (C.9). Notice that for each component C, we have

$$\overline{\log}(C)+\overline{\log}(w_{C_0})\le\overline{\log}(C_0)+\overline{\log}(C_0)$$

where C_0 is the last special component above C. Since the length of each non-special path is at most $s_{\max}$, we can bound (C.8) by

$$\widetilde{O}\Big(s_{\max}\cdot\sum_{C_0\in\mathcal{SC}}n^2\,\overline{\log}(C_0)\Big)=\widetilde{O}\Big(\sum_{C_0\in\mathcal{SC}}n^2\,\overline{\log}(C_0)\Big) \tag{C.10}$$

where $\mathcal{SC}$ is the set of all the special component produced in the algorithm. Thus it suffices to prove the following lemma.

LEMMA 17. *The bound (C.10) can be bounded by (C.9).*

Before proving this lemma, we first consider a simple case where each special component C satisfies the following condition:

$$\max_{z\in C} \overline{\log}(z) = O(\min_{z\in C} \overline{\log}(z)). \tag{C.11}$$

Since $\phi(C) \in C$ and (C.11) holds, it follows that

$$\widetilde{O}(\sum_{C\in\mathcal{SC}} n^2 \overline{\log}(C)) = \widetilde{O}(\sum_{C\in\mathcal{SC}} n^2 \overline{\log}(\xi_{\phi(C)}))$$

where $\xi_{\phi(C)}$ is an arbitrary root contained in $\phi(C)$. Thus, it is easy to see that Lemma 17 holds.

In general case, condition (C.11) may not hold for all the special components. And we call a special component **nice** if it satisfies (C.11), otherwise it is **non-nice**.

Now we define a set of square annuli for later use. Denote by w_0 the width of the smallest box centered at the origin containing $\frac{5}{4}w(B_0)$ and denote $t_0 := \lfloor \log(w_0) \rfloor$ for short. Note that if B_0 is centered at the origin, we have $w_0 = \frac{5}{4}w(B_0)$. We now define $I_{t_0+1} := \emptyset$ and

$$I_i := [-\frac{1}{2^i}, \frac{1}{2^i}]w_0,$$
$$A_i := (I_i \times I_i) \setminus (I_{i+1}, I_{i+1}),$$

for $i \in \{1, \ldots, t_0\}$. Denote $w(A_i) := \frac{1}{2} \cdot \frac{w_0}{2^i}$ as the width of the square annulus A_i.

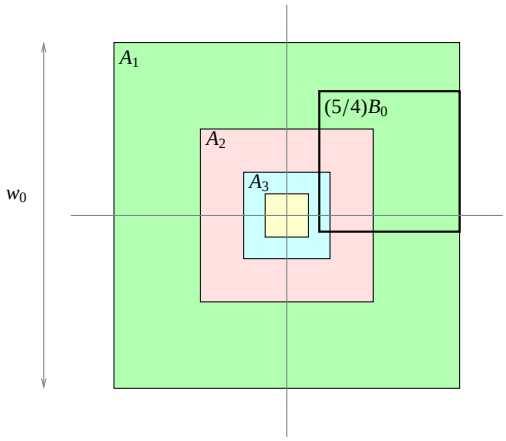

Figure C.5: Annulus A_1, A_2, A_3 and box $\frac{5}{4}B_0$.

An observation is that: for a component C, if there exists an integer $i \in \{1, \ldots, t_0 - 1\}$ such that $C \subseteq A_i \cup A_{i+1}$, then C satisfies (C.11).

Now we are prepared to prove Lemma 17.

Proof. Denote by $\mathcal{SC}_1$ the set of all the nice special components and $\mathcal{SC}_2$ the set of all the non-nice special components. From the discussions above, we can see that $\widetilde{O}(\sum_{C\in\mathcal{SC}_1} n^2 \overline{\log}(C))$ is bounded by (C.9). Thus it remains to prove that $\widetilde{O}(\sum_{C\in\mathcal{SC}_2} n^2 \overline{\log}(C))$ can be bounded by (C.9).

We define the unique set I such that $i \in I$ if and only if A_i contains at least one root in $\mathcal{Z}(Q)$. Suppose $I = i_1, \ldots, i_m$ with $i_1 < \cdots < i_m$.

We consider the components in $\mathcal{SC}_2$ that contain at least one root in A_{i_1}. Denote by $\mathcal{SC}_2(A_{i_1})$ the set of all such components and $\mathcal{Z}(A_{i_1})$ the union of the roots contained in $\mathcal{SC}_2(A_{i_1})$. We classify these components into 2 categories: the special component that contains all the roots in $\mathcal{Z}(A_{i_1})$ and the special components part of the roots in $\mathcal{Z}(A_{i_1})$. The first category consists of at most one components since any two special components contain different roots. If the first category is not empty, suppose C is the component in it. We can bound $\widetilde{O}(n^2\,\overline{\log}(C))$ with $\widetilde{O}(n^2\,\overline{\log}(B_0))$.

Now we consider the second category.

We claim that for any component C in the second category, it holds that $\overline{\log}(C) = O(\overline{\log}(w(A_{i_1})))$. The proof is as follows. We can easily see that $\mathcal{Z}(A_{i_1}) \subset B(0, 4w(A_{i_1}))$ with $B(0, 4w(A_{i_1}))$ the square centered at the origin and of width $4w(A_{i_1})$. Thus $\mathrm{rad}(\mathcal{Z}(A_{i_1})) \le 2\sqrt{2}w(A_{i_1})$. Since the second category consists of at least 2 components, thus for any component $C \in \mathcal{SC}_2(A_{i_1})$, we have $w_C \le 2 \cdot \mathrm{rad}(\mathcal{Z}(A_{i_1})) \le 4\sqrt{2}w(A_{i_1})$ (See the observation in the proof of Theorem B). Now for any $C \in P_{i_1}$, we have $\mathcal{Z}(C) \subset B(0, 4w(A_{i_1}))$ and $w_C \le 4\sqrt{2}w(A_{i_1})$. By Corollary 5(b), the distance from any point in C to a closest root in C is at most $2\sqrt{2}w_C$. Hence it is easy to see that $C \subset B(0, 4w(A_{i_1}) + 2\sqrt{2} \cdot 4\sqrt{2}w(A_{i_1})) = B(0, 20w(A_{i_1}))$. It follows $\overline{\log}(C) = O(\overline{\log}(w(A_{i_1})))$.

By the definition of $\mathcal{SC}_2(A_{i_1})$, for each component C in the second category, there exists a root contained in A_{i_1}. And since each natural ε-cluster has width less than 1, there exists a natural ε-cluster D_C in C such that $D_C \in A_{i_1} \cup A_{i_1+1}$. With the claim above, we have $\overline{\log}(C) = O(\overline{\log}(\xi_{D_C}))$ where ξ_{D_C} is an arbitrary root contained in D_C. And in this case, we charge the component C to the natural ε-cluster D_C that is contained in C. Now we prove that each D_C is charged at most $O(\log n)$ times. Suppose C' is a component in $\mathcal{SC}_2(A_{i_1})$ that is charged to D_C. Since C' is not a nice component, C' must contain a root inside A_{i_1+2}. Otherwise, C' would have satisfied the condition (C.11) since we have $\overline{\log}(C') = O(\overline{\log}(w(A_{i_1})))$. From the fact that C' contains both a root in A_{i_1} and a root inside A_{i_1+2}, we conclude that $W_{C'} \ge \frac{w(A_{i_1})}{2}$. Hence we have $w_{C'} \ge \frac{1}{3n} \cdot \frac{w(A_{i_1})}{2}$. Meanwhile, since $C' \subset B(0, 20w(A_{i_1}))$, thus $w_{C'} \le 40w(A_{i_1})$. It is easy to see that the number of different sizes of C' is bounded by $O(\log n)$. Thus we come to the conclusion that $\widetilde{O}(\sum_{C\in\mathcal{SC}_2} n^2\,\overline{\log}(C))$ is bounded by $O(\log n) \cdot \widetilde{O}(\sum_{D\in A_{i_1}\cup A_{i_1+1}} n^2\,\overline{\log}(\xi_D)) = \widetilde{O}(\sum_{D\in A_{i_1}\cup A_{i_1+1}} n^2\,\overline{\log}(\xi_D))$.

Hence we have $\widetilde{O}(\sum_{C\in\mathcal{SC}_2(A_{i_1})} n^2\,\overline{\log}(C)) = \widetilde{O}(n^2\,\overline{\log}(B_0) + \sum_{D\in A_{i_1}\cup A_{i_1+1}} n^2\,\overline{\log}(\xi_D))$.

Analogously, if we consider the components in $\mathcal{SC}_2 \setminus \mathcal{SC}_2(A_{i_1})$ that contain at least one root in A_{i_2}, we will obtain that $\widetilde{O}(\sum_{C\in\mathcal{SC}_2(A_{i_2})} n^2\,\overline{\log}(C)) = \widetilde{O}(n^2 \cdot w(A_{i+1}) + \sum_{D\in A_{i_2}\cup A_{i_2+1}} n^2\,\overline{\log}(\xi_D))$.

By recursive analysis, we can eventually obtain that the bound (C.8) is bounded by (C.9).

Q.E.D.

It remains to prove the following lemma.

LEMMA 18. *The bound (C.7) can be bounded by (C.9).*

Likewise, we first consider a simple case where each box B satisfies the following condition:

$$\max_{z\in 14B} \overline{\log}(z) = O(\min_{z\in 14B} \overline{\log}(z)). \tag{C.12}$$

Since $\phi(B) \in 14B$ and (C.12) holds, it follows that

$$\widetilde{O}(\sum_{B\in\mathcal{B}} n^2\,\overline{\log}(B)) = \widetilde{O}(\sum_{B\in\mathcal{B}} n^2\,\overline{\log}(\xi_{\phi(B)}))$$

where $\xi_{\phi(B)}$ is an arbitrary root contained in $\phi(B)$. Thus, it is easy to see that Lemma 17 holds.

In general case, condition (C.12) may not hold for all the boxes. And we call a box **nice** if it satisfies (C.12), otherwise it is **non-nice**.

Before we proving Lemma 18, we need to give a useful result.

LEMMA 19. *There exists at most* 400 *aligned non-nice boxes of the same size.*

Proof. Denote M_B as the middle of a box B. We will shows that if $M_B \notin B(O, 20w_B)$(the box centered at the origin and of width $20w_B$), then B is a nice box.

If $M_B \notin B(O, 20w_B)$, then $|M_B| > 10w_B$. We have $\min_{z\in 14B} \log(z) \geq \log(M_B - 7\sqrt{2}w_B) \geq \log(\frac{M_B}{100})$ and $\max_{z\in 14B} \log(z) \leq \log(M_B + 7\sqrt{2}w_B) \leq \log(20M_B)$. It follows $\max_{z\in 14B} \log(z) = O(\min_{z\in 14B} \log(z))$.

We can count that the number of aligned boxes satisfying $M_B \in B(0, 20w_B)$ is at most $20^2 = 400$. Thus the number of non-nice boxes of width w_B is at most 400. **Q.E.D.**

Now we prove Lemma 18.

Proof. Denote by $\mathcal{B}_1$ the set of all the nice boxes produced in the algorithm and $\mathcal{B}_2$ the set of all the non-nice boxes. From the discussions above, it follows that $\widetilde{O}(\sum_{B\in\mathcal{B}_1} n^2 \log(B))$ can be bounded by (C.9).

It remains to prove that $\widetilde{O}(\sum_{B\in\mathcal{B}_2} n^2 \log(B))$ can be bounded by (C.9). By Lemma 19, the number of non-nice boxes of the same size is at most 400. And for a box B, if B is a constituent box of a component C, it is evident that $\log(B) \leq \log(C)$. Hence $\widetilde{O}(\sum_{B\in\mathcal{B}_2} n^2 \log(B)) = 400\widetilde{O}(\sum_{C\in\mathcal{C}} n^2 \log(C)) = \widetilde{O}(\sum_{C\in\mathcal{C}} n^2 \log(C))$. By Lemma 17, the latter is bounded by (C.9). **Q.E.D.**