A Lower Bound for Computing Lagrange's Real Root Bound

Swaroop N. Prabhakar and Vikram Sharma

The Institute of Mathematical Sciences
CIT Campus, Taramani, Chennai, India 600113
npswaroop@imsc.res.in and vikram@imsc.res.in

Abstract. In this paper, we study a bound on the real roots of a polynomial by Lagrange. From known results in the literature, it follows that Lagrange's bound is also a bound on the absolute positiveness of a polynomial. A simple $O(n \log n)$ algorithm described in Mehlhorn-Ray (2010) can be used to compute the bound. Our main result is that this is optimal in the real RAM model. Our paper explores the tradeoff between improving the quality of bounds on absolute positiveness and their computational complexity.

Keywords: Real Root Bounds, Lagrange's Bound, Absolute Positiveness, Algebraic Decision Tree, Complexity Lower Bounds.

1 Introduction

Root bounds are functions that operate on univariate polynomials with complex coefficients and compute an upper bound on the absolute value of its roots. The literature contains many root bounds; see, e.g., [16, Chap. 6]. Some of these root bounds (e.g., see van der Sluis [13]), are tight relative to the largest absolute value among all the roots of the polynomial. Often, however, one is interested in the special case of upper bounds on just the positive real roots of a polynomial with real coefficients; for instance, in the continued fraction based algorithms for real root isolation [2,12]. For this special case, the literature contains some bounds [7,14,1,4,15]. In [6], Hong showed that most of the known root bounds are in fact bounds for **absolute positiveness** of a polynomial, i.e., a real number such that the polynomial and all its non-vanishing derivatives are positive for any value greater than this real number. He introduced a new bound and showed that it is tight relative to the threshold of absolute positiveness of the polynomial. The quality of a root bound is defined to be the ratio of the bound with respect to the threshold of absolute positiveness. It was shown in [4] that within a general framework of bounds on absolute positiveness, Hong's bound is nearly optimal, i.e., it is off by a constant factor with respect to the best bound that is possible in this framework. Thus in terms of quality of real root bounds, Hong's bound is nearly the best. However, it was not clear if the quality of the bound was achieved at the cost of the increased effort in computing the bound, since a naive

implementation of Hong's bound has arithmetic cost quadratic in the degree, n, of the polynomial. This computational bottleneck was overcome by Mehlhorn and Ray [9], who gave an $O(n)$ arithmetic cost algorithm to compute Hong's bound for univariate polynomials.

Recently, Collins [5] showed that a real root bound by Lagrange [8] is always better than Hong's bound. It must be noted that the Lagrange's bound had not been covered in the framework proposed in [4]. A simplified derivation of the Lagrange's bound is given in [10,3], and an extension to the complex setting is given in [10]. The improvement is by a constant factor. Given this improvement, one can ask the following questions regarding Lagrange's real root bound:

Q1. Is the bound also a bound on the absolute positiveness of the polynomial?
Q2. Can the bound be computed using $O(n)$ arithmetic operations?

In Thm. 1, we given an affirmative answer to the first question. This result is not very surprising and immediately follows from known results in the literature.

Regarding the second question, we show in Thm. 4 that the answer is negative in the real RAM model [11]. This is done by reducing a certain decision problem in geometry, called the Point-Hull Bijection problem (introduced in Sect. 4), to comparing Lagrange's real root bound with Hong's bound. We then show in Thm. 2 that the complexity of the bijection problem in the real RAM model is $\Omega(n \log n)$. This is done by showing that any algebraic decision tree for deciding the Point-Hull Bijection problem has height roughly $\Omega(n \log n)$. To obtain this lower bound, we derive a lower bound on the "topological complexity" of the bijection problem. Using standard results (see Prop. 1 and Prop. 2) these lower bounds translate to lower bounds in various computational models, in particular, the real RAM model. Therefore, the best algorithm to compute Lagrange's real root bound is essentially the $O(n \log n)$ algorithm given in [9, Sec. 3.1]. Our result highlights the tradeoff between obtaining bounds for absolute positiveness that are better in quality than Hong's bound and the arithmetic complexity of computing them. In particular, we show that the constant factor improvement in the quality of Lagrange's real root bound over Hong's bound comes at an increased computational cost. In some sense, therefore, Hong's bound attains the right compromise in this quality-vs-complexity tradeoff.

2 Absolute Positiveness of Lagrange's Real Root Bound

In this section, we will prove that the Lagrange's real root bound [8] is a bound on the absolute positiveness of a polynomial. Let

$$f(x) := x^n - \sum_{k=0}^{n-1} a_k x^k, \tag{1}$$

where $a_k \in \mathbb{R}_{\geq 0}$. Let $R(f)$ be the maximum and $\rho(f)$ be the second maximum in the sequence $|a_k|^{1/(n-k)}$, $k = 0, 1, \ldots, n-1$ (we assume that $n > 1$). **Lagrange's real root bound** of f is defined as

$$L(f) := R(f) + \rho(f). \tag{2}$$

It is known that $L(f)$ is a bound on the positive roots of f [5,10,3]. We show that it is also a bound on the positive roots of its non-vanishing derivatives. First we prove the following result, a variation of the result in [4, Lemma. 2.2], which shows that any upper bound on the positive roots of f is a bound on the absolute positiveness of f.

Lemma 1. *$L(f)$ is a bound on the absolute positiveness of f defined in (1).*

Proof. The jth derivative of f is given by

$$f^{(j)}(x) = \frac{n\,!}{(n-j)\,!}x^{n-j} - \sum_{k=j}^{n-1} \frac{k\,!}{(k-j)\,!}a_k x^{k-j}.$$

Taking $n\,!/(n-j)\,!$ common from the RHS, we get,

$$f^{(j)}(x) = \frac{n\,!}{(n-j)\,!}\left(x^{n-j} - \sum_{k=j}^{n-1} \frac{\frac{k\,!}{(k-j)\,!}}{\frac{n\,!}{(n-j)\,!}}a_k x^{k-j}\right).$$

Since $\frac{k\,!}{(k-j)\,!} < \frac{n\,!}{(n-j)\,!}$, we have

$$f^{(j)}(x) > \frac{n\,!}{(n-j)\,!}\left(x^{n-j} - \sum_{k=j}^{n-1} a_k x^{k-j}\right), \text{ for all } x > 0.$$

So,

$$f^{(j)}(x) > \frac{n\,!}{(n-j)\,!}\frac{f(x)}{x^j}, \text{ for all } x > 0.$$

Hence, $L(f)$ is a bound on the absolute positiveness of f. **Q.E.D.**

Collins [5] used $L(f)$ to improve upon a root bound due to Hong [6]. Consider a general polynomial $f(x) := \sum_{i=0}^{n} a_i x^i \in \mathbb{R}[x]$, where $a_n > 0$. For every $a_i < 0$, define

$$s_i := \operatorname{argmin}\{|a_i/a_j|^{1/(j-i)} : j > i \; a_j > 0\}. \tag{3}$$

Now for each j such that $a_j > 0$, define

$$g_j(x) := a_j x^j + \sum_{s_i = j, a_i < 0} a_i x^i.$$

Notice that g_j is in the form given in (1), so $R(g_j)$ and $\rho(g_j)$ are well-defined as the first and the second maximum, respectively, in the sequence $|a_i/a_j|^{1/(j-i)}$. Define $L(g_j)$ as in (2). However, this can be done if g_j has two or more negative coefficients; otherwise, if g_j has exactly one negative a_i, then $L(g_j)$ is taken to be the unique positive root of g_j; if g_j does not have negative coefficients, then $L(g_j) := 0$. **Lagrange's Real Root Bound of f** is defined as

$$L(f) := \max_j \; L(g_j). \tag{4}$$

To compute $L(f)$, we can compute the polynomials g_j. This can be done in $O(n \log n)$ by the algorithm given in [9, Sec. 3.1]. We can then compute $L(g_j)$ in $O(n)$ time over all j. A further linear step of computing $\max_j L(g_j)$ gives us $L(f)$. Our first result is the following:

Theorem 1. *The Lagrange Real Root Bound $L(f)$ is a bound on the absolute positiveness of f.*

Proof. Since every negative monomial $a_i x^i$ has a unique s_i associated with it, we have

$$f(x) = \sum_{a_j > 0} g_j(x).$$

From Lemma 1 and the definition of $L(g_j)$, we know that $L(g_j)$ is a bound on the absolute positiveness of g_j. Hence, from (4), we conclude that $L(f)$ is a bound on the absolute positiveness of f. **Q.E.D.**

Collins [5, Thm. 5] showed that $L(f)$ is better than the Hong's bound [6],

$$H(f) := 2 \max_{a_i < 0} \min_{\substack{a_j > 0 \\ j > i}} \left| \frac{a_i}{a_j} \right|^{1/(j-i)}.$$

Mehlhorn and Ray [9] gave an algorithm for computing $H(f)$ in $O(n)$ arithmetic operations. Can a similar algorithm exist for computing $L(f)$? In the following sections, we will answer this question in the negative.

3 Algebraic Decision Trees – Basic Notations and Definitions

Given two positive integers m, d, an (m, d)**-order algebraic decision tree** is a rooted tree T in which every internal node has associated with it a multivariate polynomial in m variables of total degree at most d. The input or domain of the decision tree is $\mathbb{R}^m$. Every internal node u of T has three children labeled "+", "–" and "0". The leaves output either a zero or a one. An algebraic decision tree computes a function from $\mathbb{R}^m$ to $\{0, 1\}$. The value of this function at $\mathbf{p} \in \mathbb{R}^m$ is computed as follows: we evaluate the polynomial associated with the root node of T at $\mathbf{p}$; depending on whether the sign of this evaluation is $-$, 0, or $+$, the computation proceeds recursively from the child of the root node labeled by the corresponding sign; we stop when we reach a leaf and output the value, either zero or one, associated with the leaf. From the description, it follows that the set of points in $\mathbb{R}^m$ that reach a given node in the tree form a semi-algebraic set. It is well known that a semi-algebraic set can be partitioned into connected components. Two points $\mathbf{p}$, $\mathbf{q} \in \mathbb{R}^m$ are said to be in the same connected component corresponding to a node u of T iff there exists a continuous curve $\gamma : [0, 1] \to \mathbb{R}^m$ such that $\gamma(0) = \mathbf{p}$, $\gamma(1) = \mathbf{q}$ and for all $t \in [0, 1]$, the point $\gamma(t)$ on the curve satisfies the set of polynomial equalities and inequalities

from the root of T to the node u. The measure of complexity in this model is the height of the decision tree T, which counts the number of worst case polynomial evaluations from the root node to a leaf.

We say that an algebraic decision tree T **solves the membership problem** for a set $S \subseteq \mathbb{R}^n$ if it satisfies the following: T outputs 1 on $\mathbf{p} \in \mathbb{R}^m$ iff $\mathbf{p} \in S$. The main idea in showing a lower bound for a membership problem in the algebraic decision tree model is to lower bound the height of T in terms of, $\#S$, the total number of connected components in the set S. We can then use the following fundamental result of Ben-Or [11, p. 102] to obtain a lower bound on the height of T:

Proposition 1. *The height of any (m, d)-order algebraic decision tree T that solves the membership problem for S is $\Omega(\log_d(\#S) - m)$.*

We will crucially use the following fact that relates lower bounds in the algebraic decision tree model with lower bounds in the real RAM model [11, p. 30].

Proposition 2. *A lower bound for a decision problem $\mathcal{A}$ in the algebraic decision tree model implies the same lower bound on $\mathcal{A}$ in the real RAM model.*

4 Lower Bound on a Geometric Problem

Consider the lower hull H of $(n+2)$ points in $\mathbb{R}^2$ such that all the $(n+2)$ points are vertices of the lower hull; note that under this assumption the vertices of H can be ordered in increasing order of x-coordinate; in this paper, we only consider such hulls. From any point $p \in \mathbb{R}^2$ there are two rays that are tangent to the hull H. Of these two rays, the lower ray from p to H is the ray such that direction of the sweep to the other ray is counterclockwise. The **lower tangent** from p to H is the line corresponding to the lower ray from p. Note that p can be on H, in which case the lower tangent is an edge containing p; in particular, if p is a vertex of H, the lower tangent is the edge that has p as the left endpoint. The **point of lower tangency** for p is the left most vertex of H on the lower tangent from p. The definition ensures that the lower tangent is well-defined for all points in the plane.

The **Point-Hull Bijection problem** is the following: For a *fixed* H, given an ordered point set $P = (p_1, \ldots, p_n)$, where $p_i \in \mathbb{R}^2$, such that *all the points in P are to the left of the leftmost vertex of H*, determine if every vertex of H, excluding the leftmost and the rightmost vertex, is a point of lower tangency for some point in P? An ordered point set P that has such a bijection to the vertices of H is called a YES-instance to the problem. All other instances of P are NO-instances; in particular, if P has a point to the right of the leftmost point of H then it is a NO-instance. Since the input is a set of n points in $\mathbb{R}^2$, we take the length of the input to be $2n$.

Known algorithms for computing the points of lower tangency test whether a given point is on one side of a given line or on the line. These tests are equivalent to evaluating a polynomial, and hence these algorithms can be modeled

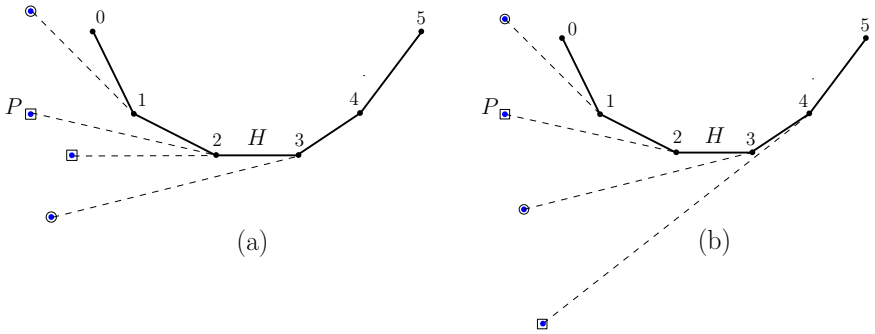

Fig. 1. A point set P shown in blue, hull H and lower tangencies. The points in P_e and P_o are shown circumscribed by boxes and circles, respectively. The figure labelled (a) is a NO-instance, whereas the figure labelled (b) is a YES-instance, to the Point-Hull Bijection problem.

as algebraic decision trees. So algebraic decision trees solving the Point-Hull bijection problem can be thought of as computing a function from $\mathbb{R}^{2n}$ to the set $\{0, 1\}$. The set of ordered point sets P that are YES-instances to the problem form a connected components in $\mathbb{R}^{2n}$. Two YES-instances are in different connected components iff all continuous paths connecting these two instances contain a NO-instance. We will now derive a lower bound on the number of such components.

Suppose P is an ordered point set that is a YES-instance to the Point-Hull Bijection problem with respect to a given hull H. By enumerating the vertices of H from left to right, starting with 0 to $(n + 1)$, we partition P into two subsets as follows:

$$P_o := \{p_i \in P \mid \ p_i\text{'s point of lower tangency on H is odd}\}$$

and

$$P_e := \{p_i \in P \mid \ p_i\text{'s point of lower tangency on H is even}\}.$$

For the ease of exposition, we assume that all the odd indices in P are in P_o and all the even indices are in P_e. We now construct a large set $\mathcal{P}$ of ordered point sets obtained from P such that all these instances are solutions to the Point-Hull Bijection problem. Keeping P_o fixed, we apply a permutation σ to the indices of points in P_e; let P_σ be the ordered point set obtained in this manner from P. Note that the permutation σ only changes the order in which the points from P_e are processed, but P_σ is still a solution to the problem. The set $\mathcal{P}$, therefore, contains $(n/2)!$ many instances that are solutions to the Point-Hull Bijection problem. We are now in a position to derive the following lower bound:

Lemma 2. *There are at least* $(n/2)!$ *connected components for the Point-Hull Bijection problem.*

Proof. Consider two distinct ordered point sets $P_\sigma, P_{\sigma'} \in \mathcal{P}$. Then we know that there is an even position $2i$ such that $j := \sigma(2i)$ is not the same as $k := \sigma'(2i)$.

In other words, the points $p_j \in P_e$ at the position indexed $2i$ in P_σ and the point $p_k \in P_e$ at the same position in $P_{\sigma'}$ are different (by construction, the points in the odd position are the same in both).

Let ℓ be the vertical line touching the leftmost point of H. Consider a continuous curve $\gamma : [0, 1] \to \mathbb{R}^{2n}$ that connects P_σ and $P_{\sigma'}$. Without loss of generality, we assume that $\gamma(t)$ stays to the left ℓ; otherwise, we obtain a NO-instance to the problem. The component, $\gamma_{2i}(t)$, of $\gamma(t)$ gives us a continuous path between p_j and p_k. Since the points in P are to the left of ℓ, and the lower tangents intersect ℓ in decreasing order of y-coordinates, it follows that the points p_j and p_k are on opposite sides of the lower tangent incident on either the $(j-1)$ or the $(j+1)$ vertex of H. As $\gamma_{2i}(t)$ is a continuous function and is also restricted to the left of ℓ, it intersects one of these tangents. So we have a point set $Q \in \mathbb{R}^{2n}$ on $\gamma(t)$ such that there are two points in Q that have the same lower tangent in H, which means that Q is a NO-instance to the problem. Therefore, P_σ and $P_{\sigma'}$ are in different connected components, and so we have the desired lower bound. For an illustration, see Figure 2.

Q.E.D.

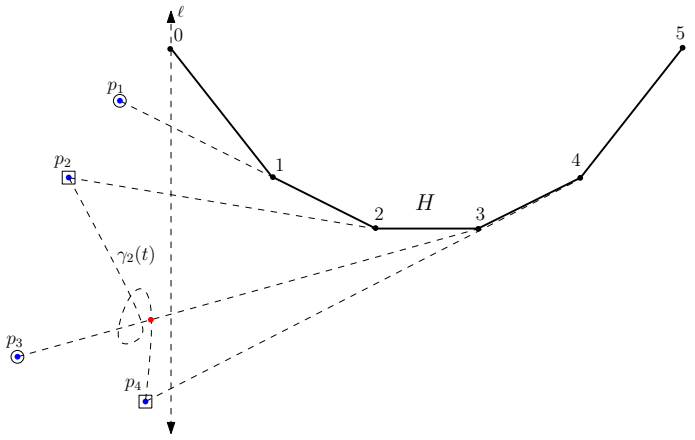

Fig. 2. In the example above $P_\sigma = \{p_1, p_2, p_3, p_4\}$ and $P_{\sigma'} = \{p_1, p_4, p_3, p_2\}$, $j = 2$ and $k = 4$. Now the component $\gamma_2(t)$ is a continuous path in $\mathbb{R}^2$ that takes p_2 to p_4. Clearly, the path intersects the lower tangent of p_3 at the point shown in red.

Using the lemma above along with Prop. 1 and Prop. 2, we obtain the following lower bound.

Theorem 2. *The arithmetic complexity of any algorithm solving the Point-Hull Bijection problem is $\Omega(n \log n)$ in the real RAM model, where $2n$ is the length of the input.*

It must be noted that d does not play a role in the lower bound above, because for a given algebraic decision tree d is fixed and hence $(1/\log d)$ is a constant.

To show the lower bound on algorithms computing $L(f)$, we need a point-hull pair that satisfies certain properties. For a hull H, let MinSlope_H and MaxSlope_H denote the least and the largest slope over the edges of H. We call a point-hull pair (P, H) **nice** if it satisfies the following conditions:

(A1): $\text{MaxSlope}_H < \text{MinSlope}_H + 1$.

(A2): The interval $(\text{MinSlope}_H, \text{MaxSlope}_H]$ contains the slopes of all the lower tangents from P to H.

(A3): The x-coordinates of points in P and H are fixed to $0, \ldots, 2n+1$.

An example of a nice point-hull pair is given in Figure 3; assumptions (A1) and (A2) are not restrictive since we can construct instances where these assumptions hold, as shown in the figure.

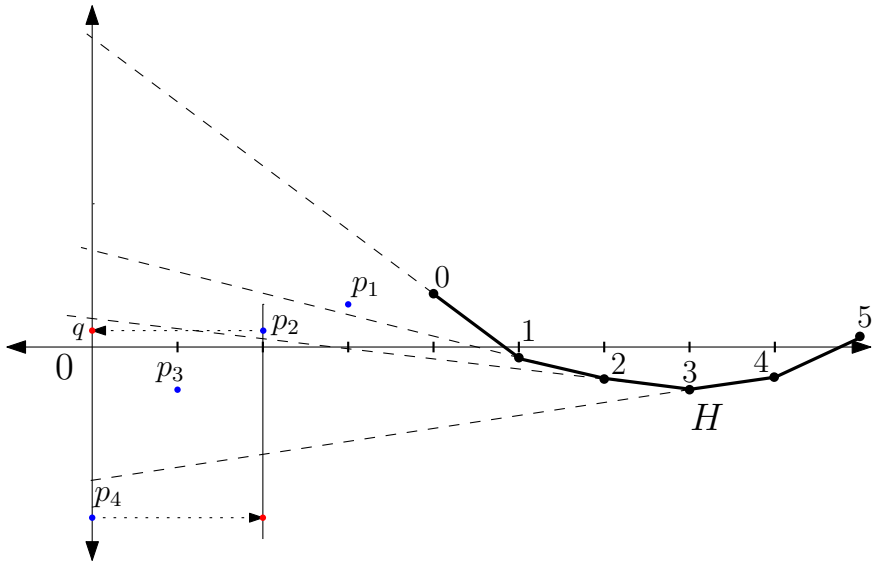

Fig. 3. The vertices labeled 0 to 5 are the vertices of H; the point set P is shown in blue; the red points are obtained by swapping the y-coordinates of p_2 and p_4. Note that q and p_3 have the same point of tangency on H.

For a nice point-hull pair (P, H), the input is only the ordered set of y-coordinates of the points in P. However, our earlier argument in Lemma 2 breaks down, because we cannot permute points in P_e, since their x-coordinates are fixed and permuting the y-coordinates may yield a NO-instance to the Point-Hull Bijection problem; e.g., in Figure 3, if we swap the y-coordinates of p_2 and p_4 then the resulting point set is a NO-instance.

For every input $\mathbf{y} \in \mathbb{R}^n$, define the ordered point set

$$P_{\mathbf{y}} := ((0, y_0), \ldots, (n-1, y_{n-1})).$$

Since the x-coordinates are fixed, we have to count the number of connected components corresponding to $\mathbf{y} \in \mathbb{R}^n$ such that $(P_{\mathbf{y}}, H)$ is a YES-instance of the Point-Hull Bijection problem.

To create a large number of input instances that are in different connected components we do the following. For $(x_i, y_i) := p_i \in P_e$, we define the following point set

$$Q_i := \{\text{points of intersection of tangents incident on even vertices in } H \text{ with the line } x = x_i\}.$$

For the example shown in Figure 3, the sets Q_2 and Q_4 are illustrated in Figure 4. For every $p_i \in P_e$, we have $|Q_i| = n/2$. So p_2 can be replaced with $n/2$ points from Q_2 corresponding to the $n/2$ tangents. However, to maintain a bijection, p_4 has to avoid the tangent on which p_2 is mapped, and so can be replaced with $((n/2)-1)$ points from Q_4. Continuing in this manner, we obtain a YES-instance $P_{\mathbf{y}'}$. The construction gives us $(n/2)!$ such input instances $\mathbf{y}' \in \mathbb{R}^n$. Our claim is that two such instances $\mathbf{y}, \mathbf{y}'$ are in different connected components in $\mathbb{R}^n$, i.e., on every continuous path connecting them there is a $\mathbf{y}''$ such that $P_{\mathbf{y}''}$ is a NO-instance to the Point-Hull Bijection problem.

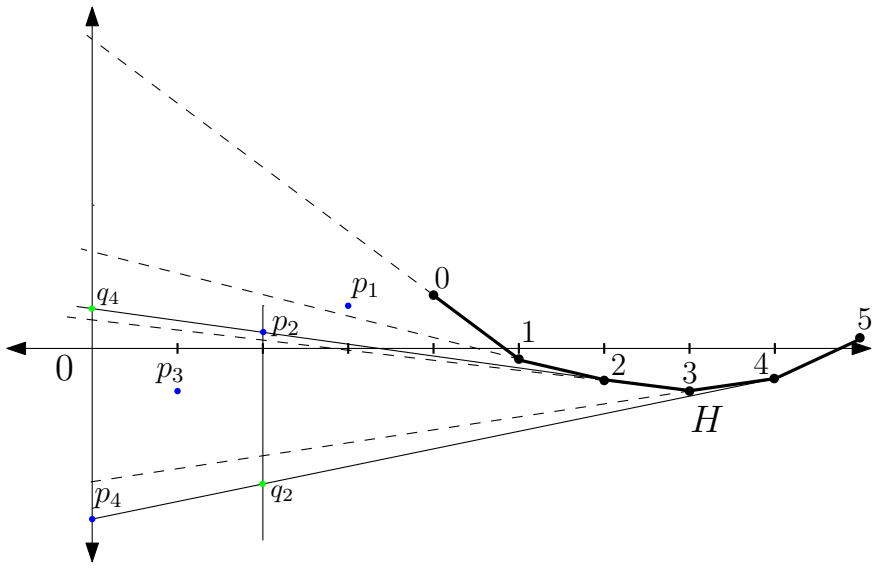

Fig. 4. The sets $Q_2 = \{p_2, q_2\}$ and $Q_4 = \{p_4, q_4\}$ corresponding to the example shown in Figure 3.

Consider a continuous curve $\gamma : [0, 1] \to \mathbb{R}^n$ connecting $\mathbf{y}$ and $\mathbf{y}'$. There has to be a $p_i \in P_e$ that is mapped to $(x_i, y_i) \in P_{\mathbf{y}}$ and $(x_i, y_i') \in P_{\mathbf{y}}'$, where $y_i \neq y_i'$. Let $\gamma_i : [0, 1] \to \mathbb{R}$ be the ith component of γ that maps y_i to y_i'. Therefore,

in $\mathbb{R}^2$, γ_i takes the point (x_i, y_i) to (x_i, y_i') along the line $x = x_i$. Since (x_i, y_i) and (x_i, y_i') are on two different tangents incident on even vertices in H, and γ_i can only move along the line $x = x_i$, it has to cross a tangent which is incident on an odd vertex of H; e.g., in Figure 4, the path from p_2 to q_2 keeping the x-coordinate fixed crosses the lower tangent corresponding to p_3. So there is a point $\mathbf{y}'' \in \mathbb{R}^n$ along the path of γ from $\mathbf{y}$ to $\mathbf{y}'$ such that $P_{\mathbf{y}''}$ is a NO-instance to the Point-Hull Bijection problem. Hence $\mathbf{y}$ and $\mathbf{y}'$ are in two different connected components in $\mathbb{R}^n$. Therefore, we apply Prop. 1 and Prop. 2, to get the following result.

Theorem 3. *The arithmetic complexity of any algorithm solving the Point-Hull Bijection problem for a nice point-hull pair (P, H) in the real RAM model is $\Omega(n \log n)$, where n is the length of the input.*

5 Lower Bound on Computing Lagrange's Real Root Bound

In this section, we will use Thm. 3 to derive a lower bound on the arithmetic complexity of computing $L(f)$ (recall the definition from (4)). Before we proceed with the derivation, we reinterpret $L(f)$.

Given a polynomial $f(x) := \sum_{i=0}^{n} a_i x^i$, let

$$p_i := (i, \log(1/|a_i|))$$

be the point corresponding to the monomial $a_i x^i$ in f. For $a_i < 0$, define s_i as in (3); recall that s_i is only defined for negative monomials. For a given p_i such that $a_i < 0$, let H_i be the lower hull of the points in the set $\{p_j : \ j > i, a_j > 0\}$. By definition of s_i we have

$$\left|\frac{a_i}{a_{s_i}}\right|^{\frac{1}{s_i - i}} = \min_{j > i; a_j > 0} \log \left|\frac{a_i}{a_j}\right|^{\frac{1}{j-i}}.$$

This can be interpreted as the slope of the lower tangent from p_i to H_i; note that if $p_j \in H_i$ is the point of lower tangency for p_i then $s_i = j$. For $a_j > 0$, define T_j as the set of lower tangents associated with p_j, i.e.,

$$T_j := \{p_i \in P, \text{ such that } s_i = j\}.$$

Let MaxSlope_{1j} and MaxSlope_{2j} be the first and second maximum over the slopes of the lower tangents of the points in T_j; if $|T_j| = 0$, then $\text{MaxSlope}_{1j} = 0$ and if $|T_j| = 1$, then $\text{MaxSlope}_{2j} = 0$. Define

$$\text{MaxSlope} := \max_j \{\text{MaxSlope}_{1j}, \text{ where } a_j > 0\}. \tag{5}$$

Then we have the following interpretations: For Hong's bound

$$H(f) = 2^{1 + \text{MaxSlope}}, \tag{6}$$

and for Lagrange's real root bound

$$L(f) = \max\left(\max_{j:\,|T_j|=1} 2^{\mathrm{MaxSlope}_{1j}}, \max_{j:\,|T_j|>1}\left(2^{\mathrm{MaxSlope}_{1j}} + 2^{\mathrm{MaxSlope}2j}\right)\right). \quad (7)$$

Using this interpretation, we will derive a lower bound on computing $L(f)$.

Theorem 4. *An algorithm for computing $L(f)$ for a real polynomial f of degree n requires $\Omega(n \log n)$ arithmetic operations in the real* RAM *model.*

Proof. The main idea of the proof is to use an algorithm for computing Lagrange's real root bound to decide the Point-Hull Bijection problem for a nice point-hull pair $(P_{\mathbf{y}}, H)$, where $\mathbf{y} \in \mathbb{R}^n$.

Let $(P_{\mathbf{y}}, H)$ be a nice point-hull pair such that

$$P_{\mathbf{y}} = \{(i, a_i) : i \in [0, \ldots, n-1], a_i \in \mathbb{R}\}$$

and

$$H = \{(i, b_i) : i \in [n, \ldots, 2n+1], b_i \in \mathbb{R}\}.$$

From $(P_{\mathbf{y}}, H)$, we construct the following polynomial

$$f(x) := \sum_{(i,b_i)\in H} \frac{x^i}{2^{b_i}} - \sum_{(i,a_i)\in P_{\mathbf{y}}} \frac{x^i}{2^{a_i}}. \quad (8)$$

This reduction from $(P_{\mathbf{y}}, H)$ to f requires $O(n)$ many exponentiation operations.

To decide the Point-Hull Bijection problem for $(P_{\mathbf{y}}, H)$, we do the following: compute $L(f)$ and $H(f)$, for f given in (8). If $2L(f) = H(f)$, we output YES; otherwise, we output NO. We now prove the correctness of this algorithm.

If $(P_{\mathbf{y}}, H)$ is a YES-instance of the Point-Hull Bijection problem, then for all j, such that $a_j > 0$, $|T_j| = 1$. Therefore, from (5), (6) and (7), we obtain that $H(f) = 2L(f)$.

Now we prove the converse: If $(P_{\mathbf{y}}, H)$ is a NO-instance of the Point-Hull Bijection problem then $2L(f) > H(f)$. Let j be an index such that $|T_j| > 1$. Then from the interpretation of $L(f)$ given in (7) we obtain that

$$\begin{aligned} 2L(f) &\geq 2(2^{\mathrm{MaxSlope}_{1j}} + 2^{\mathrm{MaxSlope}_{2j}}) \\ &\geq 2^{2+\mathrm{MinSlope}_H} \\ &> 2^{1+\mathrm{MaxSlope}_H} \\ &\geq 2^{1+\mathrm{MaxSlope}} = H(f), \end{aligned}$$

where the second and fourth inequalities follow from assumption (A2), and the third inequality follows from assumption (A1).

Since $H(f)$ can be computed with $O(n)$ many arithmetic operations, we can decide whether a nice point-hull pair $(P_{\mathbf{y}}, H)$ is a YES-instance in essentially the time taken by the algorithm for computing $L(f)$. From the lower bound in Thm. 3 and the result in [11, p. 29, Prop. 1], we get the desired claim. **Q.E.D.**

6 Conclusion and Further Directions

In this paper, we show that Lagrange's real root bound $L(f)$ is a bound on the absolute positiveness of a polynomial f. A goal in this line of work is to actually derive a tight bound on the largest positive root f, if one exists. Note that such a bound should be able to detect if f has a positive real root or not. It is clear that any algorithm for isolating real roots can be used to detect existence of a positive real root. In the converse direction, we can ask the following question: Is the problem of deciding whether a polynomial has a positive root at least as hard as isolating its real roots? One way to prove such a statement is to give a reduction from real root isolation that takes sub-quadratic (in the degree) arithmetic cost and makes at most sub-linear calls to detecting positive roots. On the other hand, one can also try to obtain an algorithm with sub-quadratic arithmetic cost for detecting or approximating positive roots.

Another direction to pursue is to generalize $L(f)$ to the multivariate setting. In [6], Hong actually derives a bound on the absolute positiveness of multivariate polynomials. In this setting, the notion of absolute positiveness is the following: A multivariate polynomial $P(x_1, \ldots, x_n)$ with real coefficients is said to be **absolutely positive** from a positive real value B iff P and all its non-zero partial derivatives of arbitrary order are positive for $x_1 \geq B, \ldots, x_n \geq B$. It is natural to derive a version of the Lagrange real root bound for multivariate polynomials and give an algorithm to compute it, similar to the one in [9]. One could then try to generalize the lower bound in Thm. 3 to this more general setting.

Acknowledgement: The authors would like to express their gratitude to Dr. Prashant Batra and the referees for their invaluable comments and suggestions.

References

1. Akritas, A.G., Strzeboński, A., Vigklas, P.: Implementations of a New Theorem for Computing Bounds for Positive Roots of Polynomials. Computing 78, 355–367 (2006)
2. Akritas, A.: Vincent's theorem in algebraic manipulation. Ph.D. thesis, Operations Research Program, North Carolina State University, Raleigh, North Carolina (1978)
3. Batra, P.: On the quality of some root-bounds. Sixth International Conference on Mathematical Aspects of Computer and Information Sciences (MACIS) (Nov, 2015), berlin, Germany
4. Batra, P., Sharma, V.: Bounds on absolute positiveness of multivariate polynomials. J. Symb. Comput. 45(6), 617–628 (2010)
5. Collins, G.E.: Krandick's Proof of Lagrange's Real Root Bound Claim. J. Symb. Comput. 70(C), 106–111 (Sep 2015), http://dx.doi.org/10.1016/j.jsc.2014.09.038
6. Hong, H.: Bounds for absolute positiveness of multivariate polynomials. J. Symb. Comput. 25(5), 571–585 (1998)
7. Kioustelidis, J.: Bounds for the positive roots of polynomials. Journal of Computational and Applied Mathematics 16, 241–244 (1986)

8. Lagrange, J.L.: Traité de la résolution des équations numériques de tous les degrés, Œuvres de Lagrange, vol. 8. Gauthier-Villars, Paris, 4th edn. (1879)
9. Mehlhorn, K., Ray, S.: Faster algorithms for computing Hong's bound on absolute positiveness. Journal of Symbolic Computation 45(6), 677 – 683 (2010), http://www.sciencedirect.com/science/article/pii/S0747717110000301
10. Mignotte, M., Ştefănescu, D.: On an Estimation of Polynomial Roots by Lagrange. Prepublication de l'Institut de Recherche Mathématique Avancée, IRMA, Univ. de Louis Pasteur et C.N.R.S. (2002), https://books.google.co.in/books?id=NAd4NAEACAAJ
11. Preparata, F.P., Shamos, M.I.: Computational Geometry: An Introduction. Springer-Verlag (1985)
12. Sharma, V.: Complexity of real root isolation using continued fractions. Theor. Comput. Sci. 409(2), 292–310 (2008)
13. van der Sluis, A.: Upper bounds for roots of polynomials. Numer. Math. 15, 250–262 (1970)
14. Ştefănescu, D.: New bounds for the positive roots of polynomials. Journal of Universal Computer Science 11(12), 2132–2141 (2005)
15. Ştefănescu, D.: A new polynomial bound and its efficiency. In: Gerdt, V.P., Koepf, W., Seiler, W.M., Vorozhtsov, E.V. (eds.) Computer Algebra in Scientific Computing - 17th International Workshop, CASC 2015, Aachen, Germany, September 14-18, 2015, Proceedings. Lecture Notes in Computer Science, vol. 9301, pp. 457–467. Springer (2015), http://dx.doi.org/10.1007/978-3-319-24021-3_33
16. Yap, C.K.: Fundamental Problems of Algorithmic Algebra. Oxford University Press (2000)