New Results in Bounds for Positiveness of Polynomials

By

Swaroop N P

MATH10201205004

The Institute of Mathematical Sciences, Chennai

A thesis submitted to the

Board of Studies in Mathematical Sciences

In partial fulfillment of requirements

for the Degree of

DOCTOR OF PHILOSOPHY

of

HOMI BHABHA NATIONAL INSTITUTE

April, 2019

Homi Bhabha National Institute

Recommendations of the Viva Voce Committee

As members of the Viva Voce Committee, we certify that we have read the dissertation prepared by Swaroop N P entitled "New Results in Bounds for Positiveness of Polynomials" and recommend that it may be accepted as fulfilling the thesis requirement for the award of Degree of Doctor of Philosophy.

V. Arvind

Chairman - Prof. V. Arvind — Date: April 3, 2019

Sharma

Guide/Convenor - Prof. Vikram Sharma — Date: April 3, 2019

C. Saha

Examiner - Prof. Chandan Saha — Date: April 3, 2019

Mmahajan

Member 1 - Prof. Meena Mahajan — Date: April 3, 2019

R. Ven

Member 2 - Prof. Venkatesh Raman — Date: April 3, 2019

Final approval and acceptance of this thesis is contingent upon the candidate's submission of the final copies of the thesis to HBNI.

I hereby certify that I have read this thesis prepared under my direction and recommend that it may be accepted as fulfilling the thesis requirement.

Date: April 3, 2019

Sharma

Place: Chennai Prof. Vikram Sharma (Guide)

STATEMENT BY AUTHOR

This dissertation has been submitted in partial fulfillment of requirements for an advanced degree at Homi Bhabha National Institute (HBNI) and is deposited in the Library to be made available to borrowers under rules of the HBNI.

Brief quotations from this dissertation are allowable without special permission, provided that accurate acknowledgement of source is made. Requests for permission for extended quotation from or reproduction of this manuscript in whole or in part may be granted by the Competent Authority of HBNI when in his or her judgement the proposed use of the material is in the interests of scholarship. In all other instances, however, permission must be obtained from the author.

Swaroop N P

DECLARATION

I hereby declare that the investigation presented in the thesis has been carried out by me. The work is original and has not been submitted earlier as a whole or in part for a degree / diploma at this or any other Institution / University.

Swaroop N P

Swaroop N P

LIST OF PUBLICATIONS ARISING FROM THE THESIS

Journal

1. **A Lower Bound for Computing Lagrange's Real Root Bound**.

Swaroop N P and Vikram Sharma.

Computer Algebra in Scientific Computing (CASC 2016).

Volume 9890 of **Lecture Notes in Computer Science (LNCS)**, pages 444–456.

2. **Improved Bounds on Absolute Positiveness of Multivariate Polynomials**.

(Extended version of our ISSAC 2017 paper)

Communicated.

Conferences

1. **Improved Bounds on Absolute Positiveness of Multivariate Polynomials**.

Swaroop N P and Vikram Sharma.

Proceedings of the International Symposium on Symbolic and Algebraic Computation (ISSAC 2017). Pages 381–388.

2. **Stronger Tradeoffs for Orthogonal Range Querying in the Semigroup Model.**

Swaroop N P and Vikram Sharma.

Proceedings of the 38th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2018).

Swaroop N P

DEDICATIONS

Dedicated to Lord Narayana...

योगस्थः कुरु कर्माणि सङ्गं त्यक्त्वा धनञ्जय ।।
सिद्ध्यसिद्ध्योः समो भूत्वा समत्वं योग उच्यते ।।४८।।

By being established in Yoga, O Dhananjaya (Arjuna), undertake actions by casting off attachment and remaining equipoised in success and failure. Equanimity itself is called Yoga (Bhagavadgita - 2.47).

ACKNOWLEDGEMENTS

I believe gratitude is the greatest virtue human beings can cultivate. As for this thesis, I have many people to whom I need to be grateful to. The foremost among them is my adviser, Vikram. Throughout the last six years of my graduate school, Vikram has been a constant source of support in every manner possible. Vikram's role in my development has been immense. During the initial years of PhD when the going was tough, it was his patience and encouragement that kept me afloat. On the technical side, Vikram has always emphasized the need to keep the larger picture in mind while trying to be precise with the finer details. Also, in matters related to technical writing and technical talks, Vikram has always offered thoughtful and sage advice. In fact, on so many occasions Vikram has lead by example by demonstrating the quality of job that one is expected to carry out. If I imbibe and cultivate even half of what Vikram has tried to pass on to me, I think that will hold me in good shape in the future.

I am deeply thankful to Arvind, Meena and Venkatesh for agreeing to be a part of my doctoral committee. During the various DC committee meetings, it was their thoughtful questions and advice that opened new directions for me in my work. My special thanks to all the members of faculty - Arvind, Vikram, Meena, Saket, Jam, Kamal, Venkatesh and CRS. Had it not been for the courses they offered, I would have probably not even opted for a career in Theoretical Computer Science.

I am also indebted to Dr. Prashant Batra for his time and assistance with technical matters related to a couple of our papers. I would also like to thank Dr. Amritanshu Prasad (Amri) for steadfastly standing by me when I was going through a personal crisis.

Next, I would like to thank Vasan (Srinivasan) for all his assistance with computer and internet related issues. At IMSC I have never faced bureaucratic hurdles and that is largely

due to the efforts of Indra madam.

The role of a strong personal support system cannot be overemphasized during PhD. One of the strongest pillars of my support system has been my friends - Anantha, Praful, Ramanath, Anuj, Prakruthi, Prasanna and Sudhin (my childhood friend).

At last and by no stretch of imagination the least, my infinite gratitude to my family - Anna, Amma, Swarna, Suma, Krishnanand, Pavan, Abhay, Anvita and Anay. It is they who have had to regularly put up with my eccentricities and unconventional choices. If not for their patience, I wouldn't have come this far. I owe it to them. I really do.

Contents

Synopsis

This thesis is divided into two parts. In the first part, we deal with the question of proving upper bounds on the positiveness of univariate and multivariate polynomials. In the second part, we concentrate on lower bounds on the problem of orthogonal range querying.

In the first part of the thesis, we begin with the problem of proving bounds on the positiveness of univariate polynomials with real coefficients. For a univariate polynomial, an upper bound on its positiveness is a positive number $\mathscr{B}$ such that the polynomial is non-negative at every value greater than or equal to $\mathscr{B}$. Assuming that the leading monomial is positive, any upper bound on the largest positive root of a univariate polynomial is also an upper bound on its positiveness. One of the well known bounds in literature is due to Hong [25]. Not only was the bound in [25] qualitatively better than the previously known bounds but it was also linear time computable [30]. Using a root bound due to Lagrange [31], in [15], Collins gave an improvement over the bound in [25]. This improved bound due to Collins is the central theme in Chapter 2. We first show that the improved bound in [15] is not only an upper bound on the largest positive root of the polynomial but also an upper bound on the largest positive root of its derivatives. Although the bound due to Collins is an improvement over Hong's bound, it wasn't known if Collins' bound admits a linear time algorithm like Hong's bound [30]. We answer this question in the negative by showing a super linear lower bound on the computation of Collins' improved bound in the real RAM model. Our lower bound on the computation of Collins' improved bound demonstrates that the bound in [25] achieves an optimal tradeoff between quality and computational

complexity. Also, our lower bound matches the best known upper bound for computing the improved bound due to Collins.

Then, we address the question of deriving upper bounds on the positiveness of multivariate polynomials. For a multivariate polynomial, a bound on its positiveness is a positive real number $\mathscr{B}$ such that the polynomial is non-negative at every point whose every coordinate is greater than or equal to $\mathscr{B}$. For multivariate polynomials, we derive a bound that improves upon the best known bound in the literature [25]. This improved bound is achieved by first generalizing a root bound due to Westerfield [49] to the multivariate setting. As a specific case of this generalized Westerfield bound, we derive a generalization of Lagrange's real root bound for multivariate polynomials. Then, we quantify the improvement of the generalized Westerfield bound and the generalized Lagrange bound over Hong's bound in the multivariate setting. Finally, we give an algorithm for computing the generalized Lagrange bound whose running time matches the running time of the best known algorithm for computing Hong's bound [35].

In the second part of the thesis, we concentrate on strengthening a lower bound on orthogonal range querying due to Fredman [22]. Our algorithm for computing the generalized Lagrange bound for multivariate polynomials performs orthogonal range querying by building range trees. So, the question of lower bound on range querying is motivated from the analysis of our algorithm to compute the generalized Lagrange bound. The problem of range querying is as follows: Given a set X of m points in n dimensions, pre-process X into a data structure such that for every query of the form $Y \in \mathbb{R}^n$, the set

$$\{X_i \in X : X_i \leq Y\}$$

can be output efficiently. Data structures for range querying typically store certain *canonical subsets* of X such that the output to every query is a disjoint union over these canonical subsets. Fredman showed that in order to prove a lower bound on range querying in the semigroup model, it suffices to prove a lower bound on the tradeoff between the sizes of the

canonical subsets and the total number of canonical subsets required to cover all the outputs. More specifically, Fredman showed that for any data structure that supports range querying on a multidimensional grid, either the total size of the canonical sets is large or the total number of canonical sets required for covering all the outputs is large. Our result shows that this tradeoff can be strengthened; *both the total size of the canonical sets and the total number of canonical sets required for covering all the outputs are large*. Our second result is an alternate proof of Fredman's tradeoff in the one dimensional setting. The problem of answering range queries using canonical subsets can be formulated as factoring a specific boolean matrix as a product of two boolean matrices, one representing the canonical sets and the other capturing the appropriate disjoint unions of the former to output all possible range queries. In this formulation, we can ask what is an optimal data structure, i.e., a data structure that minimizes the sum of the two parameters mentioned above, and how does the balanced binary search tree compare with this optimal data structure in the two parameters? The problem of finding an optimal data structure is a non-linear optimization problem. In one dimension, Fredman's result implies that the minimum value of the objective function is $\Omega(m \log m)$, which means that at least one of the parameters has to be $\Omega(m \log m)$. We show that both the parameters in an optimal solution have to be $\Omega(m \log m)$. This implies that balanced binary search trees are near optimal data structures for range querying in one dimension. We derive intermediate results on factoring matrices, not necessarily boolean, while trying to minimize the norms of the factors, that may be of independent interest We believe that our proof reveals more insight into range querying in one dimension by relating the lower bounds to the spectrum of a certain special matrix.

List of Figures

Chapter 1

Introduction

This thesis is divided into two main parts. In the first part, we study the problem of computing upper bounds on the positiveness of polynomials. Motivated by a question from the first part, in the second part, we focus on a certain lower bound on range querying due to Fredman [22]. The motivation and the background for each of the two parts along with our results will be given in the following sections.

1.1 Upper Bounds on the Positiveness of Polynomials

One of the important problems in computer algebra is to give efficiently computable upper bounds on the roots of a univariate polynomial. A special case of this problem aims to upper bound the largest positive root of the polynomial. Such bounds have algorithmic applications in root isolation and approximation [42, 4]. Before we give more details about the problem of upper bounding positive roots, we will describe briefly, the history of polynomial equations.

1.1.1 A Brief History of Polynomial Equations

[1] Polynomial equations have been one of the oldest and perhaps the most studied object in algebra for centuries. The main goal was to come up with closed formulas in terms of the coefficients of the polynomial to express its roots.

The earliest known people to solve polynomial equations were the ancient Egyptians and Babylonians. Babylonians were able to find the roots of linear and quadratic equations. One of the fundamental problems in Babylonian algebra was to find a number which when added to its reciprocal yields a given number, i.e.,

$$x + \bar{x} = b \text{ such that } x\bar{x} = 1.$$

The two equations above effectively leads to the quadratic equation $x^2 - bx + 1 = 0$. Thus, the Babylonians had the quadratic formula. Since Babylonians didn't have negative numbers, negative roots were neglected. Also, they were able to approximate the square roots of numbers. The early Egyptians knew how to solve linear equations in one variable. The process of solving though was purely arithmetical and there was no formal explanation for these arithmetical methods. Egyptian algebra was also capable of handling simple quadratic equations of the form $ax^2 = b$.

The next big development in solving polynomial equations came from Greece in the 5th century BC. A group of mathematicians known as Pythagoreans proved that square roots appearing in solutions to quadratic polynomials can give irrational numbers. The Greeks used geometrical designs made with ruler and compass to solve polynomials of degree 1, 2 and 3. In around 300 BC, Euclid developed a geometric approach for solving quadratic equations. Euclid, though did not have the notion of equation, coefficients etc. but worked purely with geometric quantities. Later, Alexandrian mathematicians, the Hero of Alexandria and Diophantus expanded on these geometric ideas which eventually

[1] The historical facts mentioned here have been taken from the book "Mathematical Thought from Ancient to Modern Times" by Morris Kline and [48].

reached the Arab world and the middle east.

In the Arab world, the task of finding roots of polynomials came to be known as "the science of restoration and balancing". The Arab word for restoration "al-jabru" is the root of the word "algebra". Al-Khwarizmi in the 9th century wrote a book on Arabic algebra and provided the fundamentals for basic algebraic theory. Before Al-Khwarizmi, the Indian mathematician Brahmagupta (5th century) had given the notation where abbreviations were used for unknowns. By medieval times, Islamic mathematicians were able to discuss the importance of the unknown variable x. They were able to multiply, divide, and find the roots of polynomials and they started to put together binomial theorems. The Persian mathematician Omar Khayyam showed how to approximate the roots of cubic equations through line segments of intersected conic sections, but was unable to come up with an expression for roots of cubic polynomials. However, in the early 13th century, Leonardo Fibonacci achieved a close approximation of the cubic equation $x^3 + 2x^2 + cx = d$.

Moving forward, in the 15th century, much progress in solving polynomial equations came from Italy. Scipio del Ferro, a professor of mathematics at Bologna had managed to solve several cubic polynomials of the form $x^3 + mx = n$. This work of Del Ferro was never published as rivals were often challenged to solve the same problems at that time. Later, Niccolo Fontana of Brescia, popularly known as Tartaglia, rediscovered the solutions to cubic equations of the form $x^3 + mx = n$. Furthermore, Fontana had also solved cubic polynomials of the form $x^3 + mx^2 = n$. Gerolamo Cardano in his work Ars Magna, then gave a solution for polynomials of degree 4 by reducing them to solving polynomials of degree 3.

A significant development with regard to the roots of polynomials was the development of the Newton's method for approximating roots of polynomials. This was published in the book "Method of Fluxions" in the 17th century. In the 18th century, it was J.L. Lagrange who greatly influenced the theory of polynomials and building on his work, Gauss proved the celebrated "Fundamental Theorem of Algebra". Gauss had also conjectured that

solving quintic equations by means of radicals might be impossible. Around the early part of the 18th century, Paolo Ruffini had almost shown that there cannot be an algebraic solution to degree 5 polynomials. But, as discovered later, Ruffini's proof was incomplete. The shortcomings in Ruffini's argument was overcome by the Norwegian mathematician N.H. Abel who conclusively showed that general polynomials of degree 5 cannot be solved by means of radicals. But, an important question was to understand the conditions under which such polynomial equations admit an algebraic solution. This question was answered by Galois in an unpublished manuscript. This manuscript was communicated to the mathematical world only a decade later after Galois' death by Liouville.

Due to the negative results of Ruffini and Abel, mathematicians have sought efficient methods for approximating the roots of polynomials. Some examples of such methods are the Newton iteration method, Graeffe's iteration etc. Fourier devised a root approximation scheme which involved isolating the roots first and then approximating them to the desired precision. Many algorithms based on Descartes' rule of signs, Budan's theorem, Vincent's theorem and Sturm sequences have been proposed for root isolation. Building on Vincent's theorem, Akritas [4] gave an algorithm for root isolation based on the idea of continued fractions. A key sub-task in continued fractions method for isolation of real roots is to compute an upper bound on the largest positive root of a certain polynomial. It is in this context that upper bounds on the real roots of a polynomial are important.

1.1.2 Positiveness of Polynomials

For a univariate polynomial with real coefficients, a bound on **positiveness** is a positive real number $\mathscr{B}$ such that the polynomial is non-negative at every value which is greater than or equal to $\mathscr{B}$. Many problems in mathematics, logic etc., can be reduced to testing positiveness of a polynomial over reals. Hence the notion of positiveness is an important concept. Assuming that the leading monomial is positive, an *upper bound on the largest positive root of the polynomial is an upper bound on the positiveness of the polynomial*.

The problem of computing upper bounds on the absolute value of the roots of a polynomial has a long standing history with some of the bounds being attributed to Cauchy and Lagrange [31]. Root bounds are functions that operate on univariate polynomials with complex coefficients and compute an upper bound on the absolute value of its roots. For some well known root bounds see [50, Chap. 6] and [37, p. 144]. Some of these root bounds (e.g., see van der Sluis [47]), 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 example, in the continued fraction based algorithms for real root isolation [4, 42]. For this special case, the literature contains some bounds [29, 44, 3, 7, 45, 40, 6, 48]. Among these known bounds, the bound due to Kioustelidis [29] is very well known and was widely used till Hong [25] introduced a better bound. Hong's bound is in fact a bound on *absolute positiveness*, i.e., it is not only a bound on the positiveness of the polynomial but also a bound on the positiveness of its non-vanishing derivatives. In [25], it was also shown that most of the known root bounds are in fact bounds for absolute positiveness of a polynomial. The notion of absolute positiveness is relevant since for many polynomials, a bound on the positiveness of the polynomial does not imply a bound on the positiveness of its derivatives. The **quality** of a root bound is defined to be the ratio of the bound with respect to the threshold of absolute positiveness. The smaller this ratio is, the better the bound. In [23], it was shown that most upper bounds for positive roots can be arbitrarily bad, i.e., their ratio with respect to the largest positive root of the polynomial can approach infinity. Also, in [23], it was proved that under certain conditions, the ratio of Hong's bound with respect to the largest positive root is off by a factor which is at most linear in the degree of the polynomial. From [7], we also see 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. But, how does Hong's bound perform with regard to **computational complexity**? This is an important question due to

the fact that root bounds have algorithmic applications such as in root isolation algorithms as already mentioned. Ideally, we would like to compute a root bound in time linear in the degree of the polynomial. But, a naive implementation of Hong's bound has arithmetic cost quadratic in the degree, d, of the polynomial. This computational bottleneck was overcome in [30], where an $O(d)$ arithmetic cost algorithm to compute Hong's bound for univariate polynomials was given.

Recently, Collins [15] showed that a real root bound by Lagrange [31] 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 [7]. A simplified derivation of the Lagrange's bound is given in [36, 5, 15] and an extension to the complex setting is given in [36]. The improvement is by a constant factor. A slight modification of the algorithm in [30] can be used to compute Collins' improvement in $O(d \log d)$ time. Given this improved bound, one can ask the following questions regarding the Lagrange real root bound:[2]

Q1. Is the bound also a bound on the absolute positiveness of the polynomial?

Q2. Can this improved bound be computed using $O(d)$ arithmetic operations?

We address the questions Q1 and Q2 in Chapter 2. Our results are:

1. In Theorem 2, we show that the Lagrange real root bound is an upper bound on the absolute positiveness of the polynomial, i.e., not only does it upper bound the positive roots of the polynomial but also its derivatives.

2. In Theorem 10, we prove a $\Omega(d \log d)$ lower bound on the computation of the Lagrange real root bound in the real RAM model. Our lower bound matches the best known upper bound on the computation of the Lagrange real root bound.

The problem of proving upper bounds on positiveness generalizes naturally for multivariate polynomials. We say a positive real number $\mathscr{B}$ is an upper bound on the positiveness

[2] We refer to Collins' improved bound [15] as the "Lagrange real root bound" since Collins uses Lagrange's bound in improving upon Hong's bound.

of a multivariate polynomial $F(x_1, x_2, \ldots, x_n) \in \mathbb{R}[x_1, \ldots, x_n]$ if the polynomial F is non-negative at every point whose every coordinate is at least as large as $\mathscr{B}$, i.e.,

$$F(x_1, x_2, \ldots, x_n) \geq 0, \text{ for all } x_i \geq \mathscr{B}.$$

A well known bound on the positiveness of multivariate polynomials is due to Hong [25]. Hong's bound for multivariate polynomials and the question of improving upon it is the central theme of Chapter 3 of this thesis. More specifically, we concentrate on the question of generalizing Lagrange's real root bound for multivariate polynomials and showing that it is an improvement over Hong's bound. In generalizing Lagrange's bound, we first generalize a root bound due to Westerfield [49] for univariate polynomials to the multivariate setting. Since Westerfield's bound is more generic than the Lagrange bound in the univariate setting, a generalization of Westerfield's bound gives us a generalization of Lagrange's bound. To understand this, we first illustrate the difference between the various bounds in the univariate setting: Consider the following special polynomial $x^d - \sum_{i=0}^{d-1} a_i x^i$, where $a_i \geq 0$. For this special polynomial, Hong's bound is $2\max_i a_i^{1/(d-i)}$, whereas Lagrange's bound [31] is the sum of the first and the second maximum in the radical sequence $a_i^{1/(d-i)}$, $i = 0, \ldots, d-1$. To obtain both the bounds for a general univariate polynomial, we decompose it as a sum of such special polynomials and take the maximum over the bound for each polynomial in the decomposition. Westerfield's bound is more generic in the sense that it is obtained by taking a suitable positive linear combination of all the values in the radical sequence $a_i^{1/(d-i)}$, $i = 0, \ldots, d-1$, and therefore is an improvement over both Hong's and Lagrange's bound. So, in Chapter 3, we address the following questions:

Q3. What are the generalizations of Lagrange's and Westerfield's bound to the multivariate setting?

Q4. Are these generalizations better than Hong's bound for multivariate polynomials?

Q5. Can these generalizations be computed as effectively as Hong's bound?

Our main results in Chapter 3 are:

1. In Theorem 15, we give a generalization of Westerfield's root bound to the multivariate setting. In Section 3.2.1, we derive a generalization of Lagrange's real root bound as a specific case of the generalized Westerfield bound for multivariate polynomials.

2. In Lemma 16, we quantify the improvement of Westerfield's bound and Lagrange's bound over Hong's bound in the multivariate setting.

3. In Section 3.3, we give an algorithm to compute the Lagrange bound for multivariate polynomials. Then, in Theorem 17, we show that the running time of this algorithm matches the running time of the Mehlhorn-Ray algorithm for computing Hong's bound [35] when $n \geq 2$. In the univariate setting, such an algorithm to compute the Lagrange real root bound whose running time matches the running time of the algorithm to compute Hong's bound is not possible. This is due to our $\Omega(d \log d)$ lower bound on the computation of the Lagrange real root bound in Chapter 2.

1.1.3 Lower Bound on Orthogonal Range Querying

Chapter 4 of the thesis is devoted to lower bounds on the problem of orthogonal range querying. Our algorithm for computing the multivariate Lagrange bound in Chapter 3 performs orthogonal range querying. In order to understand the efficiency of our algorithm with respect to data structures for orthogonal range querying, we refer to a known lower bound of Fredman [22] on range querying. In Chapter 4, we focus on strengthening this lower bound on orthogonal range querying.

Orthogonal range querying is one of the fundamental problems in computational geometry. The **range querying** problem is the following: Given a set X of m points in $\mathbb{R}^n$ and a *range* set $\mathscr{R}$ of subsets of points in $\mathbb{R}^n$, the goal is to pre-process the set X into a data structure so that given a query range $R \in \mathscr{R}$, the set of points in $X \cap R$ can be output

efficiently. For **orthogonal range querying**, a range is simply an axis aligned box in $\mathbb{R}^n$. In this thesis, we only consider the problem of orthogonal range querying. Sometimes, we are also interested in the number of points in the set $X \cap R$. The case where we output all the points in $X \cap R$ is called range reporting and the case where we only report the number of points in $X \cap R$ is called range counting. Other types of queries include whether or not $X \cap R$ is empty and so on. To capture these different types of queries in the range querying framework, it is typical to associate with every point $X_i \in X$ a weight w_i, where w_i comes from a semigroup $(S,+)$[3]. Then, for every query range R, the output is $\sum_{X_i \in X \cap R} w_i$. For instance, for the orthogonal range reporting problem, we can take the semigroup $(2^X, \cup)$ and set $w_i = \{X_i\}$; for the range counting problem, we can take the semigroup $(\mathbb{N},+)$ and set $w_i = 1$.

Data structures for range querying typically store certain **canonical subsets** of the input set X and on a query range R, the query algorithm comes up with a set of disjoint canonical subsets such that their union is exactly $X \cap R$. The performance of a data structure for range querying is measured by the time spent in answering a query, the space requirement of the data structure and also the preprocessing cost involved in building the data structure. Often, the preprocessing is ignored as the data structure is built only once. In the dynamic setting where operations such as delete and insert are permitted, update time is also important. Most data structures for geometric problems are described in the real RAM model [41] and the pointer-machine model [1, 2]. A popular data structure for orthogonal range querying is the *range tree* which was introduced by Bentley [9]; for an exposition, see [10, chap. 5]. For orthogonal range reporting on m points in n dimensions, the range tree can be built in time $O(m \log^{n-1} m)$ and every query can be answered in time $O(\log^n m + k)$, where k is the number of points in the output. The query time though can be improved to $O(\log^{n-1} m + k)$ through a technique called fractional cascading [14, 33]. These upper bounds have been subsequently improved for range querying in various computation models [1, 2].

[3] Another algebraic structure from which weights are assigned are groups [21, 19], but in this thesis we restrict ourselves to the case where weights come from a semigroup.

Fredman gave some of the first lower bounds on orthogonal range querying in the semigroup model [20, 22]. These lower bounds are in the dynamic setting where insertions and deletions are allowed. More specifically, in [22], he showed that for any m, there is a sequence of m operations consisting of insert, delete and querying such that the time required for this sequence is $\Omega(m \log^n m)$ in the semigroup model. Intuitively, at the heart of this lower bound argument lies a certain tradeoff; if all the queries are easy to answer, i.e., the querying algorithm needs to select very few canonical sets to answer a query, then the points that occur very frequently in the outputs must be stored in many canonical sets. In this case, a query to delete or insert a point that occurs in many outputs will be expensive. On the other hand, if the querying algorithm needs to select many canonical sets to answer a query, then the time spent in choosing the required canonical sets is too much. This suggests an inherent tradeoff between the cost of updating the data structure and the cost of querying. So, the required lower bound is obtained by balancing the cost of an update (insertion or deletion) operation and the cost of a querying operation. In some sense, the cost of update depends on the number of canonical subsets that a given element is stored in. On the other hand, the cost of querying depends on the sizes of canonical subsets. For instance, if the data structures stores all the outputs itself as the canonical sets, then the querying algorithm needs to select only the appropriate canonical set. This hints at a tradeoff between the sizes of the canonical sets and the number of canonical sets needed to answer all the query ranges. This tradeoff is what Fredman exploits in showing the required lower bound [34, p. 69, Lemma 9]. More precisely, Fredman showed that for any data structure that supports range querying, **either** the sum of the sizes of the canonical sets is large **or** the number of canonical sets needed to answer all the query ranges is large. Motivated by the analysis of our algorithm for computing the Lagrange bound for multivariate polynomials in Section 3.3.3, we prove a stronger version of this combinatorial tradeoff: that **both** the sum of the sizes of the canonical sets and the number of canonical sets needed to answer all the query ranges are large. Then, in Section 4.2, we show that the balanced binary search tree is near optimal for range querying in one

dimension in a more relaxed framework.

Chapter 2

A Lower Bound on Computing Lagrange's Real Root Bound

In this chapter, we will present our results[1] on the Lagrange real root bound for univariate polynomials. Our main results in this chapter are:

1. In Theorem 2, we show that the Lagrange real root bound is a bound on the absolute positiveness of the polynomial. The proof of this theorem involves partitioning an arbitrary polynomial as a sum of some special polynomials and then arguing that the Lagrange bound of these special polynomials is a bound on their absolute positiveness(See Lemma 1 and Theorem 2 for details).

2. With regard to the question of computational complexity of the Lagrange real root bound, we show that there does not exist a linear time algorithm for computing the Lagrange real root bound. More specifically, in Theorem 10, we show an $\Omega(d \log d)$ lower bound on the computation of the Lagrange real root bound in the real RAM model. In order to prove a lower bound in the RAM model, we follow the usual approach of proving a lower bound in the *algebraic decision tree* model which then implies the same lower bound in the real RAM model. In proving the

[1] Our results in this chapter were published in [39].

required lower bound on the computation of the Lagrange real root bound, we first interpret the Lagrange real root bound in a geometric setting. Then by using this geometric interpretation, we formulate a decision problem and then show a lower bound on this problem in the algebraic decision tree model(see Theorem 9). Then, in Theorem 10, we show a reduction from the geometric decision problem to the problem of computing the Lagrange real root bound to get the desired lower bound.

In the next section, we will define the problem of root bounds formally and give a description of some well known bounds in the literature.

2.1 Upper Bounds on the Positiveness of Univariate Polynomials

From hereon in this chapter, we will assume that the **leading monomial** of a polynomial is positive. Given a real polynomial

$$g(x) := \sum_{i=0}^{d} a_i x^i, \quad a_i \in \mathbb{R}$$

the numbers $\alpha \in \mathbb{R}$ such that $g(\alpha) = 0$, are referred to as the **real roots** of g.

DEFINITION 1. *An upper bound on the positiveness of a polynomial $g(x) \in \mathbb{R}[x]$ is a positive real number $\mathscr{B}$ such that*

$$g(x) \geq 0, \textit{ for all } x \geq \mathscr{B}.$$

Since the leading monomial of g is assumed to be positive, any upper bound on the positive real roots of g is an upper bound on the positiveness of g. Hence, to obtain a bound on the positiveness of a polynomial, it suffices to upper bound its largest positive root.

DEFINITION 2. *$\mathscr{B}$ is referred to as a bound on the absolute positiveness of g if*

- *$\mathscr{B}$ is an upper bound on the positiveness of g and*
- *$\mathscr{B}$ is also an upper bound on the positiveness of its non-vanishing derivatives.*

A bound on absolute positiveness need not be the same as the bound on the positiveness of a polynomial. For instance, the polynomial $(x-1)^2$ is positive from 0 onwards but its derivative $2(x-1)$ is positive from 1.

Note that by computing an upper bound on the positive roots of $g(-x)$, we can obtain an upper bound on the negative roots of $g(x)$. In this chapter, we will be concerned only with upper bounds on the positive roots and for the sake of succinctness, we will refer to them as just **root bounds**.

We will now describe the characteristics of a good root bound:

1. Quality: It is the ratio of the bound with respect to the infimum of all bounds on the absolute positiveness of the polynomial. This infimum is referred to as the *threshold of absolute positiveness. The smaller the ratio, the better the bound.*
2. *Computational complexity: Since root bounds have algorithmic applications, we would like them to efficiently computable. Ideally, we would like to have an algorithm whose running time is linear in the degree of the polynomial for computing a given root bound.*

We will now present some root bounds in the literature which are relevant to our main results in this chapter: Consider a polynomial

$$g := \sum_{i=0}^{d} a_i x^i \in \mathbb{R}[x].$$

B1. A well known and a frequently used real root bound is due to Kioustelidis [29],

$$K(g) := 2 \max_{a_i<0} \left| \frac{a_i}{a_d} \right|^{1/(d-i)}.$$

In some sense, Kioustelidis' bound tries to account for the contribution from the negative monomials by pairing them only with the leading monomial. The contribution of the remaining positive monomials is not considered. From the definition of $K(g)$, it is clear that it can be computed in time linear in the degree of the polynomial. For the implementation details of the algorithm, we refer to [48, p. 21]. Here we note that this bound has also been independently shown by Johnson [28].

B2. A more recent bound due to Hong [25] improves upon Kioustelidis' bound by considering the contribution of all the positive monomials. Hong's bound,

$$H(g) := 2 \max_{a_i<0} \min_{\substack{a_j>0 \\ j>i}} \left| \frac{a_i}{a_j} \right|^{1/(j-i)}. \tag{2.1}$$

With regard to quality, Hong showed that $H(g)$ is off by a constant factor of d from the threshold of absolute positiveness. In [7], it has been argued that $H(g)$ is near optimal within a certain framework. Coming to computational complexity, Hong's bound is linear time computable due to an algorithm in [30].

A bound asserted by Lagrange [31] happens to be better than Kioustelidis' bound. Collins [15] showed that this bound of Lagrange can be used to improve upon Hong's bound. This improvement of Collins over Hong's bound is the central theme of this chapter. In the next section, we will define Lagrange's bound and show that it is a bound on the absolute positiveness of a polynomial.

2.2 Absolute Positiveness of Lagrange's Real Root Bound

In this section, we will prove that Lagrange's real root bound [31] is a bound on the absolute positiveness of a polynomial. Let

$$f(x) := x^d - \sum_{k=0}^{d-1} a_k x^k, \tag{2.2}$$

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

$$L(f) := R(f) + \rho(f). \tag{2.3}$$

It is known that $L(f)$ is a bound on the positive roots of f [15, 36, 5]. 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 [7, 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. *[39] $L(f)$ is a bound on the absolute positiveness of f defined in* (2.2).

Proof. The j-th derivative of f is given by

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

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

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

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

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

So,

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

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

The bound in (2.3) can be generalized to an arbitrary polynomial as follows: Consider

a **general polynomial**

$$g(x) := \sum_{i=0}^{d} a_i x^i \in \mathbb{R}[x], \tag{2.4}$$

where $a_d > 0$. For every $a_i < 0$, define

$$s_i := \operatorname{argmin}\{|a_i/a_j|^{1/(j-i)} : j > i,\ a_j > 0\}. \tag{2.5}$$

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

$$g_j(x) := a_j x^j + \sum_{s_i = j, a_i < 0} a_i x^i. \tag{2.6}$$

Notice that g_j is in the form given in (2.2), 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 (2.3). 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 $L(g_j) := 0$. **The Lagrange Real Root Bound of** g is defined as[2]

$$L(g) := \max_j L(g_j). \tag{2.7}$$

To compute $L(g)$, we can compute the polynomials g_j. This can be done in time $O(d \log d)$ by the algorithm given in [35, Sec. 3.1]. We can then compute $L(g_j)$ in $O(d)$ time over all j. A further linear step of computing $\max_j L(g_j)$ gives us $L(g)$. Our first result is the following:

THEOREM 2. *[39] The Lagrange Real Root Bound $L(g)$ is a bound on the absolute*

[2]This bound is the improved bound of Collins. Since the Lagrange bound is used for improvement, we call it the "Lagrange real root bound"

positiveness of g.

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

$$g(x) = \sum_{a_j>0} g_j(x) + h(x)$$

where $h(x)$ is a sum of positive monomials. 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 (2.7), we conclude that $L(g)$ is a bound on the absolute positiveness of g. **Q.E.D.**

Collins [15, Thm. 5] showed that $L(g)$ is better than the Hong's bound in (2.1). Mehlhorn and Ray [35] gave an algorithm for computing $H(g)$ in $O(d)$ arithmetic operations. Can a similar algorithm exist for computing $L(g)$? In the following sections, we will answer this question in the negative. We begin by giving an overview of the model of computation in which we show a lower bound on the computation of the Lagrange real root bound.

2.3 Algebraic Decision Trees - Basic Notations and Definitions

In this section, we will give an overview of the model of computation in which we prove a lower bound on the computation of the Lagrange real root bound. Consider the following classical decision problem of element distinctness: Are any two numbers in the sequence $(x_1, x_2, \ldots, x_d)$ equal? A simple solution to this problem is to sort the sequence and then scan the sorted list to find out if there are any duplicates. The time required by this solution is $O(d \log d)$ since we need to sort the input. Intuitively, this simple solution also seems like the best solution possible. But, how do we prove an $\Omega(d \log d)$ lower bound? Since this is a decision problem with only a 'yes' or a 'no' answer, the information theoretic

argument used for the lower bound on sorting in the comparison tree model does not work. This motivates the need for a stronger model of computation.

The simplest type of decision trees are the **linear decision trees** introduced by Dobkin and Lipton [17]. In a linear decision tree, every internal node is labeled with a vector $(a_0, a_1, \ldots, a_d)$ and has three outgoing edges labeled $+$, $-$ and 0. Given an input $(x_1, x_2, \ldots, x_d)$, at every node the following linear polynomial is evaluated:

$$a_0 + \sum_{i=1}^{d} a_i x_i.$$

Depending on the sign of the polynomial evaluation above, we decide which node to branch to. The leaves of this tree are labeled either 0 or 1 and the label of the leaf at which the computation ends is taken as the output of the tree. So, linear decision trees naturally compute a function $F : \mathbb{R}^d \to \{0,1\}$ and every internal node defines a hyperplane. The measure of the complexity in this model is the **depth** of the tree which is a count of the number of polynomial evaluations.

To prove lower bounds, we use the geometric interpretation of linear decision trees: Consider the set of all points S_v in $\mathbb{R}^d$ that reach a given leaf v in the tree. All the points in S_v satisfy a set of linear equalities and inequalities thereby forming a convex polyhedron. Hence, the set S_v is connected. This property of connectivity gives us a nice handle to prove lower bounds. Dobkin and Lipton [17] showed that

LEMMA 3. *The depth of any linear decision tree computing a function $F : \mathbb{R}^d \to \{0,1\}$ is $\Omega(\log(\#F_0 + \#F_1))$ where $\#F_0$ is the number of connected components of points x such that $F(x) = 0$ and $\#F_1$ is defined similarly.*

Hence if we want to prove a lower bound on a problem in the linear decision tree model, we only need to bound the corresponding $\#F_0$ and $\#F_1$. The general flavour of the lower bound argument can be explained with the following example: Again, consider the element distinctness problem where the input is $X := (x_1, x_2, \ldots, x_d) \in \mathbb{R}^d$. Assuming X is a 'yes'

instance, i.e., elements of X are distinct, we permute the elements of X to generate $d!$ 'yes' instances. Now the idea is to argue that these $d!$ instances are all in different connected components thereby obtaining a lower bound on $\#F_1$. To see this, consider any two distinct permutations of X, say, U, V. There exists indices i, j such that $U_i < U_j$ and $V_i > V_j$. Now, any continuous path $\gamma : [0,1] \to \mathbb{R}^d$ such that $\gamma(0) = U$ and $\gamma(1) = V$ must contain a point W such that $W_i = W_j$. Clearly, W is 'no' instance of the element distinctness problem. Now by using the connectedness property of the points reaching a given node in the tree, we can conclude that U and V must be in different connected components.

An obvious generalization of the linear decision tree model was given by Steele and Yao [43]: **Algebraic decision trees**. The only difference between a linear decision tree and an algebraic decision tree is that the linear polynomials in the internal nodes of the linear decision tree are replaced with degree n multivariate polynomials. Formally, an algebraic decision tree is defined as follows: Given two positive integers m, n, an (m,n)**-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 n. The input or domain of the decision tree is $\mathbb{R}^m$ and the function from $\mathbb{R}^m$ to $\{0,1\}$ is computed exactly in the same manner as in the case of linear decision trees. However, there is one crucial difference between inputs to a linear decision tree and inputs to an (m,n)-order algebraic decision tree, T: the set of points in $\mathbb{R}^m$ that reach a given node in the tree T form a semi-algebraic set and not a convex polyhedron. 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 again the **depth** 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 $S \subseteq \mathbb{R}^m$ if it satisfies the following: T outputs 1 on $\mathbf{p} \in \mathbb{R}^m$ iff $\mathbf{p} \in S$. However, in this model we cannot leverage the connectedness argument as in the linear decision tree model to prove lower bounds. This is because semi-algebraic sets can be disconnected. But, the following lower bound on the number of connected components turns out to be useful:

PROPOSITION 4. *[27, 38] Let X be a semi-algebraic set in $\mathbb{R}^m$ defined by k polynomial equalities and h polynomial inequalities and the degree of all polynomials is at most $n \geq 2$. Then, the number of connected components of X is upper bounded by $n(2n-1)^{m+h-1}$*

Ben-Or [8], using the lower bound above on the number of connected components showed that

PROPOSITION 5. *The height of any (m,n)-order algebraic decision tree T that solves the membership problem for S is $\Omega(\log_n(\#S) - m)$, where $\#S$ is the number of connected components in S.*

As in the case of linear decision trees, the crux of the lower bound argument in the algebraic decision tree model boils down to lower bounding the number of connected components in the required semi-algebraic set.

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 [41, p. 30].

PROPOSITION 6. *A lower bound for a decision problem $\mathscr{A}$ in the algebraic decision tree model implies the same lower bound on $\mathscr{A}$ in the real RAM model.*

We would also like to note that the real RAM model that we consider allows the following operations:

- The arithmetic operations.
- Comparisons between two real numbers.

- kth root, trigonometric functions, logarithmic function, exponentiation and other analytic functions.

2.4 Lower Bound on a Geometric Problem

We start this section by motivating the definition of a certain geometric problem which will be used for our lower bound argument.

2.4.1 Motivation

In order to obtain a lower bound on the computation of the Lagrange real root bound in (2.7), we need to understand the geometric idea that underlies the linear time algorithm of Koprowski-Mehlhorn-Ray [30] for the computation of Hong's bound. For an arbitrary polynomial $g := \sum_{i=0}^{d} a_i x^i \in \mathbb{R}[x]$, recall the Hong's bound of g in (2.1). From the definition of $H(g)$, we see that for every negative monomial in $a_i x^i$ in g, Hong's bound tries to associate it with a unique positive monomial $a_j x^j$ such that $j > i$ and minimizes the ratio $\left|a_i/a_j\right|^{1/(j-i)}$. This 'association' can be understood geometrically as follows: By taking the logarithm of the ratio $\left|a_i/a_j\right|^{1/(j-i)}$,

$$\log \left|\frac{a_i}{a_j}\right|^{\frac{1}{(j-i)}} = \frac{\log|a_i| - \log|a_j|}{j-i},$$

we see that the RHS of the equality above is the **slope** of the line passing through the points $(i, -\log|a_i|)$ and $(j, -\log|a_j|)$. So, for a negative monomial $a_i x^i$, the minimum of the ratios $\left|a_i/a_j\right|^{1/(j-i)}$, for all $j > i$ and $a_j > 0$, is the slope of the **lower tangent** from the point $(i, -\log|a_i|)$ to the **lower hull** of the **dominating set** of points

$$\left\{(j, -\log|a_j|) : j > i, a_j > 0\right\}. \tag{2.8}$$

With the interpretation above, we see that computing $H(g)$ is equivalent to computing the maximum of the slope of such lower tangents. This geometric interpretation of Hong's bound is what the linear time algorithm in [30] leverages.

Coming to the Lagrange real root bound, $L(g)$ can be computed by computing the polynomials g_j defined in (2.6). Computation of g_j's requires computation of s_i as defined in (2.5) for every negative monomial. From the definition of s_i in (2.5), we see that s_i is the **point of lower tangency** from the point $(i, -\log|a_i|)$ to the lower hull of its dominating set as defined in (2.8). In this algorithm, computation of s_i's is the expensive step which requires construction of lower hulls and needs time $O(d \log d)$. The reason for this logarithmic blowup in running time is due to the way in which the negative monomials are processed. For Hong's bound, we are only interested in computing the maximum of a certain quantity, namely the slope of the lower tangents from points corresponding to negative monomials to the lower hull of their dominating sets. So, if the slope corresponding to the negative monomial being processed is not larger than some precomputed slope, then this negative monomial can be discarded. But, we cannot do this for the computation of the Lagrange bound, $L(g)$. This is due to the fact that $L(g)$ is the maximum of $L(g_j)$ and $L(g_j)$ is defined as the sum of $R(g_j)$ and $\rho(g_j)$ as defined in (2.3). We note that the quantities $R(g_j)$ and $\rho(g_j)$ are simply the exponentiation of the first and the second maximum of the slopes of points whose points of tangency is $(j, -\log|a_j|)$. Therefore, while computing $L(g)$, we cannot discard a negative monomial whose s_i is j unless we are sure that the Lagrange bound of the associated g_j will not give us the bound $L(g)$. So in some sense, the worst case input for the computation of $L(g)$ is the polynomial g that forces the algorithm to compute the lower tangents corresponding to all the negative monomials. This is the crucial aspect which our lower bound argument exploits. In the next section, we will formulate a geometric decision problem based on computing lower tangents from a point set to a lower hull. Then, we will show a lower bound on this decision problem. By interpreting the points as monomials, we will show that if there is a linear time algorithm for the computation of the Lagrange real root bound,

then there is an algorithm for the geometric decision problem that violates the lower bound shown for it.

2.4.2 The Point-Hull Bijection Problem

Consider the lower hull H of $(d+2)$ points in $\mathbb{R}^2$ such that all the $(d+2)$ 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 chapter, 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 $P = (p_1, \ldots, p_d)$, 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 d points in $\mathbb{R}^2$, we take the length of the input to be $2d$.

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 as algebraic decision trees.

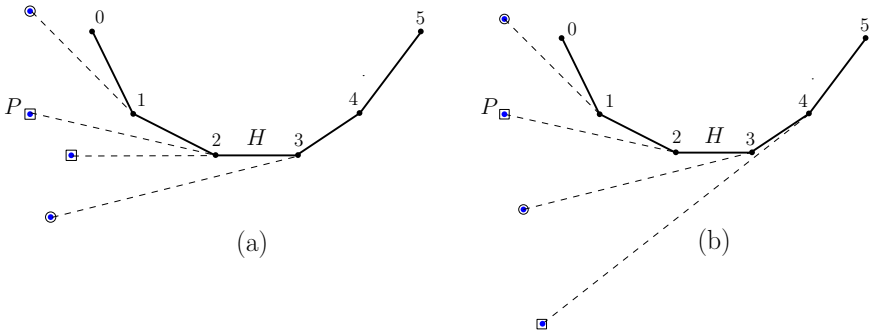

Figure 2.1: A point set P shown in blue, hull H and lower tangencies. The points in P_e and P_o are shown circumscribed by boxes and circles, respectively. The figure labelled (a) is a NO-instance, whereas the figure labelled (b) is a YES-instance, to the Point-Hull Bijection problem.

So algebraic decision trees solving the Point-Hull bijection problem can be thought of as computing a function from $\mathbb{R}^{2d}$ to the set $\{0,1\}$. The set of ordered point sets P that are YES-instances to the problem form a connected components in $\mathbb{R}^{2d}$. 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 $(d+1)$, we partition P into two subsets as follows:

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

and

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

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 $\mathscr{P}$ 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 YES-instance of the problem. The set $\mathscr{P}$, therefore, contains $(d/2)!$ many instances that are solutions to the Point-Hull Bijection problem. We are now in a position to derive the following lower bound:

LEMMA 7. *There are at least* $(d/2)!$ *connected components for the Point-Hull Bijection problem.*

Proof. Consider two distinct ordered point sets $P_\sigma, P_{\sigma'} \in \mathscr{P}$. 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 $\gamma : [0,1] \to \mathbb{R}^{2d}$ 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 $Q \in \mathbb{R}^{2d}$ 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 2.2.

Q.E.D.

Using the lemma above along with Proposition 5 and Proposition 6, we obtain the following lower bound.

THEOREM 8. *The arithmetic complexity of any algorithm solving the Point-Hull Bijection problem is* $\Omega(d \log d)$ *in the real RAM model, where* $2d$ *is the length of the input.*

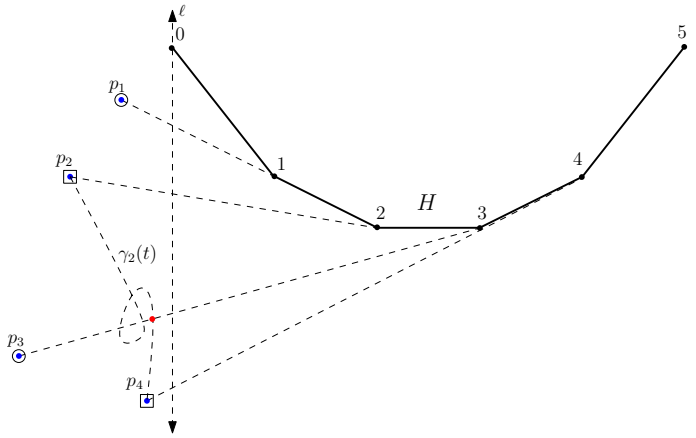

Figure 2.2: In the example above $P_\sigma = \{p_1, p_2, p_3, p_4\}$ and $P_{\sigma'} = \{p_1, p_4, p_3, p_2\}$, $j = 2$ and $k = 4$. Now the component $\gamma_2(t)$ is a continuous path in $\mathbb{R}^2$ that takes p_2 to p_4. Clearly, the path intersects the lower tangent of p_3 at the point shown in red.

It must be noted that n in Proposition 5 does not play a major role in the lower bound above, because for a given algebraic decision tree n is fixed and hence $(1/\log n)$ is a constant.

To show the lower bound on algorithms computing $L(g)$ where g is as defined in (2.4), 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 $0, \ldots, 2d+1$.

An example of a nice point-hull pair is given in Figure 2.3; assumptions (A1) and (A2) are not restrictive since we can construct instances where these assumptions hold, as shown in the figure.

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 7 breaks down, because we cannot

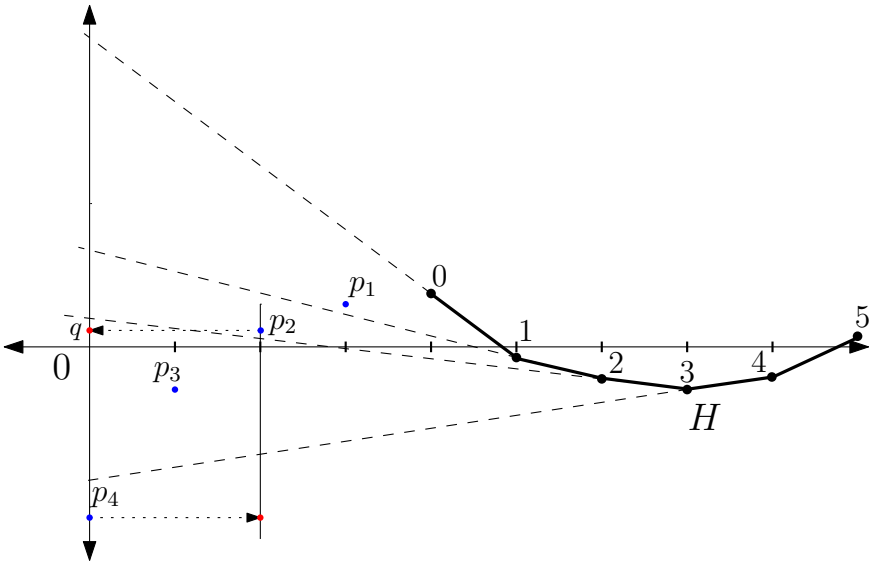

Figure 2.3: The vertices labeled 0 to 5 are the vertices of H; the point set P is shown in blue; the red points are obtained by swapping the y-coordinates of p_2 and p_4. Note that q and p_3 have the same point of tangency on H.

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 2.3, if we swap the y-coordinates of p_2 and p_4 then the resulting point set is a NO-instance.

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

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

Since the x-coordinates are fixed, we have to count the number of connected components corresponding to $\mathbf{y} \in \mathbb{R}^d$ 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 2.3, the sets Q_2 and Q_4 are illustrated in Figure 2.4. For every $p_i \in P_e$, we have $|Q_i| = d/2$. So p_2 can be replaced with $d/2$ points from Q_2 corresponding to the $d/2$ 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 $((d/2)-1)$ points from Q_4. Continuing in this manner, we obtain a YES-instance $P_{\mathbf{y}'}$. The construction gives us $(d/2)!$ such input instances $\mathbf{y}' \in \mathbb{R}^d$. Our claim is that two such instances $\mathbf{y}, \mathbf{y}'$ are in different connected components in $\mathbb{R}^d$, 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.

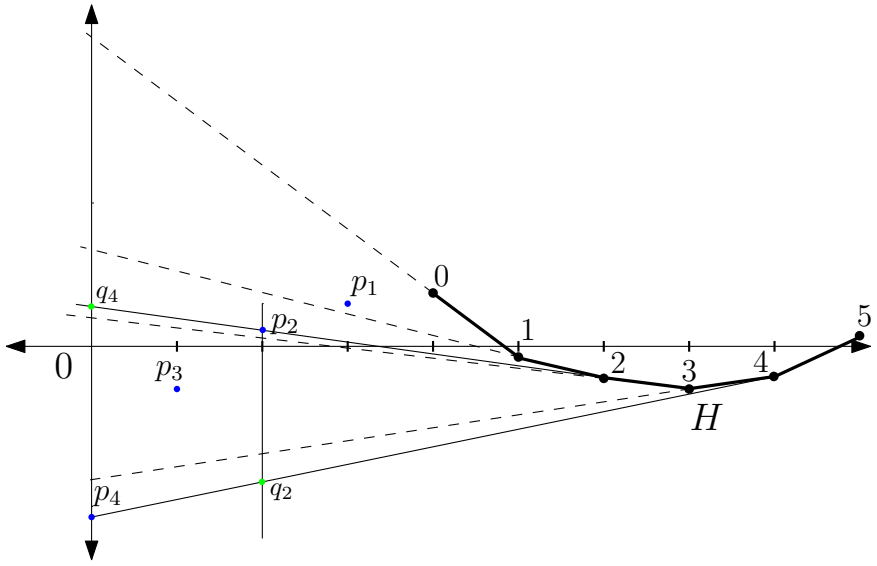

Figure 2.4: The sets $Q_2 = \{p_2, q_2\}$ and $Q_4 = \{p_4, q_4\}$ corresponding to the example shown in Figure 2.3.

To see this, consider a continuous curve $\gamma : [0,1] \to \mathbb{R}^d$ 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 2.4, 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 $\mathbf{y}'' \in \mathbb{R}^d$ 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 $\mathbb{R}^d$. Therefore, we apply Proposition 5 and Proposition 6, to get the following result.

THEOREM 9. *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 $\Omega(d \log d)$, where d is the length of the input.*

2.5 Lower Bound on Computing Lagrange's Real Root Bound

In this section, we will use Theorem 9 to derive a lower bound on the arithmetic complexity of computing $L(g)$ (recall the definition from (2.7)). Before we proceed with the derivation, we recall the interpretation of $L(g)$ in the geometric setting.

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

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

be the point corresponding to the monomial $a_i x^i$ in g. For $a_i < 0$, define s_i as in (2.5); 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 dominating set as defined in (2.8). 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

$$\text{MaxSlope} := \max_j \left\{ \text{MaxSlope}_{1j}, \text{ where } a_j > 0 \right\}. \tag{2.9}$$

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

$$H(g) = 2^{1+\text{MaxSlope}}, \tag{2.10}$$

and for Lagrange's real root bound

$$L(g) = \max\left(\max_{j:\, |T_j|=1} 2^{\text{MaxSlope}_{1j}}, \max_{j:\, |T_j|>1} \left(2^{\text{MaxSlope}_{1j}} + 2^{\text{MaxSlope}2j} \right) \right). \tag{2.11}$$

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

THEOREM 10. *An algorithm for computing $L(g)$ for a real polynomial g of degree d requires $\Omega(d \log d)$ arithmetic operations in the real* RAM *model.*

Proof. The main idea of the proof is to use an algorithm for computing the Lagrange 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

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

and

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

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

$$f(x) := \sum_{(i,b_i) \in H} \frac{x^i}{2^{b_i}} - \sum_{(i,a_i) \in P_{\mathbf{y}}} \frac{x^i}{2^{a_i}}. \tag{2.12}$$

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

To decide the Point-Hull Bijection problem for $(P_{\mathbf{y}}, H)$, we do the following: compute $L(g)$ and $H(g)$, for g given in (2.12). If $2L(g) = H(g)$, 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 (2.9), (2.10) and (2.11), we obtain that $H(g) = 2L(g)$.

Now we prove the converse: If $(P_{\mathbf{y}}, H)$ is a NO-instance of the Point-Hull Bijection problem then $2L(g) > H(g)$. Let j be an index such that $|T_j| > 1$. Then from the interpretation of $L(g)$ given in (2.11) we obtain that

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

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

Since $H(g)$ can be computed with $O(d)$ 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 $L(g)$. From the lower bound in Theorem 9 and the result in [41, p. 29, Prop. 1], we get the desired claim. **Q.E.D.**

Here we note that the algorithm for computing Hong's bound in [30] assumes that the monomials are given to us in the increasing order of their degree. But our lower bound on the computation of the Lagrange real root bound is independent of any such assumption and hence, it is applicable to any algorithm for computing the Lagrange real root bound.

2.6 Conclusion

In this chapter, we have shown that the Lagrange real root bound, $L(g)$, is a bound on the absolute positiveness of a univariate real polynomial g. A goal in this line of work is to actually derive a tight bound on the largest positive root of g, if one exists. Note that such a bound should be able to detect if g 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 $L(g)$ to the multivariate setting. In [25], a bound on the absolute positiveness of multivariate polynomials is given. 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 [35]. One could then try to generalize the lower bound in Theorem 10 to this more general setting. In the next chapter, we will focus on generalizing the Lagrange real root bound $L(g)$ for multivariate polynomials.

Chapter 3

Generalizing the Lagrange Real Root Bound for Multivariate Polynomials

In this chapter[1], we will present our results on upper bounds on the absolute positiveness of multivariate polynomials. Our main results are:

1. In (3.21), we give a generalization of Westerfield's bound for multivariate polynomials. Then, in Theorem 15, we show that it is a bound on the absolute positiveness of multivariate polynomials. In proving Theorem 15, we first express an arbitrary multivariate polynomial as a sum of special polynomials as defined in (3.2) and then show that the Westerfield bound of these special polynomials is a bound on their absolute positiveness (see (3.19) and Lemma 13). Also, in (3.25), we derive a generalization of the Lagrange real root bound for multivariate polynomials. Then, in Lemma 16, we quantify the improvement of the generalized Westerfield bound and the generalized Lagrange bound over Hong's bound in the multivariate setting.

2. For computing the Lagrange bound for multivariate polynomials, we give an algorithm in Section 3.3. In Theorem 17, we show that our algorithm matches the running time of the Mehlhorn-Ray algorithm [35] for computing Hong's bound. A

[1]The results given in this chapter appeared in [40].

key aspect of our algorithm for computing the Lagrange bound is that it uses range trees for range querying. By referring to a known lower bound on range querying due to Fredman [22], in Section 3.3.3, we argue that the running time of our algorithm cannot be improved by using a different data structure for range querying.

3.1 Preliminaries

Consider a univariate real polynomial, $f(x) := \sum_{i=0}^{d} a_i x^i$, $a_d > 0$. From (2.3), Lagrange's real root bound of f is

$$L(f) := R(f) + \rho(f),$$

where $R(f)$ and $\rho(f)$ are the first and the second maximum respectively in the sequence

$$\left(|a_i/a_d|^{1/(d-i)} : a_i < 0 \right).$$

As mentioned in chapter 2, Collins [15] showed that $L(f)$ can be used to improve upon a real root bound in [25]. It is natural to ask whether the bound can be further improved if we consider more numbers from the sequence $\left(|a_i/a_d|^{1/(d-i)} : a_i < 0 \right)$. This approach has been explored by Westerfield [49] for deriving a root bound, i.e., an upper bound on the absolute value of roots of a complex polynomial. For a monic complex univariate polynomial $z^d + \sum_{i=0}^{d-1} a_i z^i$, he considers its Cauchy polynomial $z^d - \sum_{i=0}^{d-1} |a_i| z^i$ and defines

$$b_i := \text{the } i\text{th maximum in the sequence } \left(|a_i|^{1/(d-i)} \right).$$

Westerfield then defines a sequence of polynomials, $(P_k)_{k \in \mathbb{N}}$, where

$$P_k(y) := y^k - \sum_{j=0}^{k-1} y^j,$$

whose unique positive roots are in increasing order, i.e., if y_k is the positive root of P_k, then

$$1 = y_1 < y_2 < \cdots < y_k < \cdots,$$

and in the limit they converge to 2. Let $\beta_1 := 1$ and for $k > 1$ define

$$\beta_k := y_k - y_{k-1}.$$

Therefore, for any $n \in \mathbb{N}$, we have $1 \leq \sum_{k=1}^{n} \beta_k < 2$. Westerfield proved the following root bound [49]:

PROPOSITION 11 (Westerfield'33). *If $\alpha \in \mathbb{C}$ is a root of the polynomial $z^d + \sum_{i=0}^{d-1} a_i z^i$, then*

$$|\alpha| \leq \sum_{i=1}^{d} \beta_i b_i =: W(f).$$

In the next section, we generalize Westerfield's bound to a bound on the absolute positiveness in the multivariate setting. Before that, we fix the following notation for this chapter:

Notations:

1. For $\mu := (\mu_1, \ldots, \mu_n) \in \mathbb{Z}^n_{\geq 0}$, define $|\mu| := \mu_1 + \mu_2 + \cdots + \mu_n$.

2. Given $\mu, \nu \in \mathbb{Z}^n_{\geq 0}$, define $\nu - \mu := (\nu_1 - \mu_1, \ldots, \nu_n - \mu_n)$.

3. For $\mu, \nu \in \mathbb{Z}^n_{\geq 0}$, we introduce the partial ordering " $\preceq$ " and write $\mu \preceq \nu$ if for all $i \in [n]$, $\mu_i \leq \nu_i$. We write $\mu \prec \nu$, if $\mu \preceq \nu$ and $\mu \neq \nu$.

4. Given $\mu \in \mathbb{Z}^n_{\geq 0}$, define $\mu! := \mu_1! \cdots \mu_n!$.

5. Let $X := (x_1, \ldots, x_n)$ be an n-tuple of real variables; we will use x to denote a univariate real variable. For a $\mu \in \mathbb{Z}^n_{\geq 0}$, define $X^\mu := x_1^{\mu_1} \cdots x_n^{\mu_n}$.

6. Given $\lambda \in \mathbb{Z}^n_{\geq 0}$, and a polynomial $F(X) \in \mathbb{R}[X]$, its partial derivative of order λ,

$$F^{(\lambda)}(X) := \frac{\partial^{|\lambda|} F}{\partial x_1^{\lambda_1} \cdots \partial x_n^{\lambda_n}}.$$

7. Given $B \in \mathbb{R}$, by $X \geq B$ we mean $x_1 \geq B, \ldots, x_n \geq B$.

8. Let

$$\Omega_n := 1 - \sqrt[n]{\frac{1}{2}}. \tag{3.1}$$

It is known that $1/\Omega_n \geq \frac{n}{\ln 2} + 0.5 = 1.44n + 0.5$, and not hard to verify that Ω_n is a root of the series $2 - (1-x)^{-n}$.

The following definition generalizes the notion of a real root bound to the multivariate setting [25]:

DEFINITION 3. *A real number B is said to be a bound on the* **positiveness** *of $F \in \mathbb{R}[X]$ if*

$$\textit{for all } X \geq B, F(X) \geq 0.$$

Hong [25] showed that most of the known bounds are not just upper bounds on the positiveness of F but also upper bounds on the positiveness of its derivatives. Therefore, we will be interested in bounding the following quantity:

DEFINITION 4. *A real number B is said to be a bound on the* **absolute positiveness** *of $F \in \mathbb{R}[X]$ if*

1. *It is bound on the positiveness of F.*
2. *For all $X \geq B$, $F^{(\lambda)}(X) \geq 0$, for any partial derivative of F of order λ.*

It is not clear whether there exists a bound on the absolute positiveness of an arbitrary multivariate polynomial $F(X)$. To ensure that F has such a bound, we need to make

certain assumptions. In order to state the assumptions, we need the notion of a dominating monomial [25].

DEFINITION 5. *A monomial $a_\nu X^\nu$ is said to dominate a monomial $a_\mu X^\mu$ if and only if $\mu \prec \nu$. A dominating monomial is the one that is not dominated by any other monomial in the polynomial.*

To ensure the existence of a bound for absolute positiveness, Hong and Jakus [26] showed that we need the following assumptions:

1. The coefficients of all the dominating monomials in F are positive.

2. At least one of the monomials in F has a negative coefficient; otherwise, zero is a trivial bound on the absolute positiveness.

From now we assume that F satisfies these two assumptions. Without the assumption on the coefficients of the dominating monomials being positive, definition 3 is not relevant. For instance, consider the polynomial $x^2 - y^2 - 1$; in this polynomial, x^2 and y^2 are the dominating monomials. From definition 3, a bound on positiveness is a real number B such that $x^2 - y^2 - 1$ is non-negative in the orthant hinged at the point (B,B). But, clearly for any real number B, there will be infinitely many points in the orthant hinged at (B,B) where the monomial x^2 cannot dominate $(-y^2 - 1)$.

3.2 A Westerfield-type Bound in the Multivariate Setting

We first derive a bound on the absolute positiveness of the following special multivariate polynomial: given $\nu \in \mathbb{Z}^n_{\geq 0}$ define

$$F_\nu(X) := X^\nu + \sum_{0 \preceq \mu \prec \nu} a_\mu X^\mu, \tag{3.2}$$

where $a_\mu \in \mathbb{R}_{\leq 0}$, for all μ. As in Westerfield's bound for univariate polynomials, we define b_i's and B_i's with a slight difference, namely we only consider non-zero a_μ's: Define

$$b_i := \text{the } i\text{th largest number in the } \textit{sequence } \left(\left|a_\mu\right|^{1/|\nu-\mu|}, a_\mu \neq 0\right), \tag{3.3}$$

and

$$B_i := \text{the } i\text{th largest in the } \textit{set } \left\{\left|a_\mu\right|^{1/|\nu-\mu|}, a_\mu \neq 0\right\}, \tag{3.4}$$

where $0 \preceq \mu \prec \nu$. *If the number of b_i's is smaller than $|\nu|$ then we repeat the smallest amongst them to make up for the deficit.* Thus the b_i's can occur with repetition and their number is at least $|\nu|$; the B_i's, on the contrary, are the b_i's without repetition. Following Westerfield, we also define a **canonical sequence of univariate polynomials**

$$P_k(y) := y^k - \sum_{j=1}^{k} \binom{n+j-1}{j} y^{k-j},\ k \in \mathbb{N}. \tag{3.5}$$

In particular, $P_1(y) = y - n$, $P_2(y) = y^2 - ny - \binom{n+1}{2}$ and so on. As in the univariate setting, the unique positive roots of P_k's are in increasing order, i.e., if y_k is the positive root of P_k, then

$$n = y_1 < y_2 < \cdots < y_k < \cdots.$$

Analogous to the univariate setting, we define $\beta_1 := n$ and for $k > 1$

$$\beta_k := y_k - y_{k-1}. \tag{3.6}$$

The following result will be useful later:

LEMMA 12. *Let $k \in \mathbb{N}$ and $\nu \in \mathbb{Z}^n_{>0}$ be such that $|\nu| = k$. Then*

1. *For $y \geq 0$ we have*

$$P_k(y) \leq y^{|\nu|} - \sum_{0 \preceq \mu \prec \nu} y^{|\mu|}.$$

2. *The roots* $y_k < \frac{1}{\Omega_n}$. *Furthermore,*

$$\frac{1}{\Omega_n} - y_k > c_{n,k} \frac{\Omega_n^k}{k(1-\Omega_n e^{-1/k})},$$

where

$$c_{n,k} := \begin{cases} 1 & \text{for } n = 1 \\ \left(1 + \frac{k+1}{n-1}\right)^{n-1} & \text{for } n > 1. \end{cases} \tag{3.7}$$

Note that $c_{n,k} < e^{k+1}$.

Proof.

1. The claim is trivially true if $y = 0$, so we assume that $y > 0$. From the definition of P_k, it follows that the claim is equivalent to showing that

$$y^k - \sum_{j=1}^{k} \binom{n+j-1}{j} y^{k-j} \leq y^{|\nu|} - \sum_{0 \preceq \mu \prec \nu} y^{|\mu|}, \text{ for } y > 0.$$

Since $|\nu| = k$, we divide both sides by y^k, cancel the unit term on both sides, and substitute $x = 1/y$ to obtain the following equivalent claim:

$$\sum_{j=1}^{k} \binom{n+j-1}{j} x^j \geq \sum_{0 \preceq \mu \prec \nu} x^{|\nu|-|\mu|}, \text{ for } x > 0.$$

The coefficient of x^j on the RHS is equal to the number of μ's such that $|\mu| = k - j$ and $\mu \prec \nu$. Since $|\nu| = k$, the number of such μ's is at most the number of non-negative solutions to the equation $x_1 + x_2 + \cdots + x_n = j$. A standard combinatorial argument shows that the number of such solutions is bounded by $\binom{n+j-1}{j}$. Therefore, we conclude

$$P_k(y) \leq y^{|\nu|} - \sum_{0 \preceq \mu \prec \nu} y^{|\mu|}, \text{ for } y > 0.$$

2. The series $Q(x) := 2-(1-x)^{-n}$ vanishes at Ω_n. Using the identity $\binom{-n}{i} = (-1)^i\binom{n+i-1}{i}$, we obtain that

$$Q(x) = 1 - nx - \binom{n+1}{2}x^2 - \binom{n+2}{3}x^3 - \cdots.$$

But for $x \geq 0$, we have that the polynomial $x^k P_k(1/x) \geq Q(x)$. As both $x^k P_k(1/x)$ and $Q(x)$ monotonically decrease for $x \geq 0$, it follows that $\Omega_n < 1/y_k$, or equivalently $y_k < 1/\Omega_n$.

To lower bound $(1/\Omega_n - y_k)$, we will derive a lower bound on $y_{i+1} - y_i$, and take the sum of these lower bounds for $i \geq k$. In order to lower bound $y_{i+1} - y_i$, observe that the polynomial P_i is convex from y_i onwards, therefore, the Newton iterates starting from y_{i+1} converge to y_i from above. Hence, y_i is smaller than the first Newton iterate $y_{i+1} - P(y_{i+1})/P'(y_{i+1})$, that is,

$$y_{i+1} - y_i \geq \frac{P(y_{i+1})}{P'(y_{i+1})}. \tag{3.8}$$

From the definition of the polynomials P_{i+1} and P_i, it follows that

$$P_{i+1}(y) = yP_i(y) - \binom{n+i}{i+1}.$$

Evaluating both sides at y_{i+1}, and substituting the value of $P_i(y_{i+1})$ in (3.8), we obtain that

$$y_{i+1} - y_i \geq \binom{n+i}{i+1}\frac{1}{y_{i+1}\cdot P_i'(y_{i+1})}.$$

Since $P_i'(y_{i+1}) \leq i y_{i+1}^{i-1}$, the inequality above, along with the observation that $y_{i+1} < 1/\Omega_n$, implies that

$$y_{i+1} - y_i \geq \binom{n+i}{i+1}\frac{1}{i y_{i+1}^i} > \binom{n+i}{i+1}\frac{\Omega_n^i}{i}.$$

Taking the sum of the LHS for $i \geq k$, we obtain that

$$\frac{1}{\Omega_n} - y_k > \sum_{i \geq k} \binom{n+i}{i+1} \frac{\Omega_n^i}{i}. \tag{3.9}$$

The desired lower bound is essentially a slightly better estimate than the first term in the summation above. Since the binomial term increases with i, we can replace it with the binomial term corresponding to $i = k$, and then pull out the first term from the summation to get a lower bound:

$$\frac{1}{\Omega_n} - y_k > \binom{n+k}{k+1} \frac{\Omega_n^k}{k} \sum_{i \geq k} \frac{\Omega_n^{i-k}}{i/k}. \tag{3.10}$$

Now we will derive a lower bound on the summation term in the RHS above. First, we reorder the index to start from zero to obtain

$$\sum_{i \geq k} \frac{\Omega_n^{i-k}}{i/k} = \sum_{i \geq 0} \frac{\Omega_n^i}{1 + (i/k)}.$$

Since for all x, $(1+x) < e^x$, we get that

$$\sum_{i \geq 0} \frac{\Omega_n^i}{1 + (i/k)} > \sum_{i \geq 0} (\Omega_n e^{-1/k})^i = \frac{1}{1 - \Omega_n e^{-1/k}}.$$

Substituting this lower bound in (3.10) gives us,

$$\frac{1}{\Omega_n} - y_k > \binom{n+k}{k+1} \frac{\Omega_n^k}{k(1 - \Omega_n e^{-1/k})}. \tag{3.11}$$

For $n = 1$ the binomial coefficient is just one. When $n > 1$, the binomial coefficient in the inequality above is

$$\binom{n+k}{k+1} = \binom{n+k}{n-1} = \frac{(n+k)(n+k-1)\cdots(k+2)}{(n-1)(n-2)\cdots 1}.$$

In the RHS above, we substitute the lower bound

$$\frac{n+k-i}{n-1-i} \geq \frac{n+k}{n-1},$$

for $i = 0, \ldots, n-2$, to obtain that

$$\binom{n+k}{k+1} \geq \left(\frac{n+k}{n-1}\right)^{n-1} = \left(1+\frac{k+1}{n-1}\right)^{n-1}.$$

Substituting this lower bound in (3.11) gives the required inequality:

$$\frac{1}{\Omega_n} - y_k > c_{n,k}\frac{\Omega_n^k}{k(1-\Omega_n e^{-1/k})}.$$

Q.E.D.

Now we generalize Westerfield's bound for the special polynomial F_ν as defined in (3.2).

LEMMA 13. *For the polynomial F_ν given in* (3.2)*, define*

$$W(F_\nu) := \sum_{i=1}^{|\nu|} \beta_i b_i. \tag{3.12}$$

Then $W(F_\nu)$ is an upper bound on the positiveness of F_ν.

Proof. For the sake of succinctness, let $W := W(F_\nu)$. Consider,

$$\begin{aligned} F_\nu(X) &= X^\nu + \sum_{0 \preceq \mu \prec \nu} a_\mu X^\mu. \\ &= X^\nu \left(1 + \sum_{0 \preceq \mu \prec \nu} a_\mu \frac{1}{X^{\nu-\mu}}\right). \end{aligned} \tag{3.13}$$

Since $a_\mu < 0$, for every μ, the equation above implies that $F_\nu(X) \geq F_\nu(W)$, for $X \geq W$. So to prove that W is a bound on the positiveness of F_ν, it suffices to prove that $F_\nu(W) \geq 0$. Since we substitute W for each x_i in F_ν to get $F_\nu(W)$, we consider the univariate polynomial

$F_\nu(x)$ obtained by substituting $x_1 = x_2 = \cdots = x_n = x$ in F_ν, i.e., the polynomial

$$F_\nu(x) := x^{|\nu|} + \sum_{0 \preceq \mu \prec \nu} a_\mu x^{|\mu|}, \tag{3.14}$$

where $a_\mu \in \mathbb{R}_{\leq 0}$, for all μ. To show that $F_\nu(W) \geq 0$, we follow the same approach as in Westerfield's proof [49], which is by induction on the the number of non-zero B_i's.

1. Base case: There is exactly one non-zero B_i; let it be b. So,

$$F_\nu(x) = x^{|\nu|} - \sum_{0 \preceq \mu \prec \nu} b^{|\nu-\mu|} x^{|\mu|}.$$

If $|\nu| = k$, then from Lemma 12, we have that for $y \geq 0$

$$P_k(y) \leq y^{|\nu|} - \sum_{0 \preceq \mu \prec \nu} y^{|\mu|}.$$

Replacing y by x/b and multiplying by b^k in this inequality, we get that

$$b^k P_k(x/b) \leq x^{|\nu|} - \sum_{0 \preceq \mu \prec \nu} b^{|\nu-\mu|} x^{|\mu|} = F_\nu(x).$$

So the positive root of $F_\nu(x)$ is at most $by_k = b \sum_{i=1}^{|\nu|} \beta_i = W$.

2. Induction case: Let $B_1 > \ldots > B_\ell > 0$ be the $\ell > 1$ distinct non-zero values in the set $\left\{ |a_\mu|^{1/|\nu-\mu|}, 0 \preceq \mu \prec \nu \right\}$. In the non-increasing sequence of b_i's, let k_1 be the number of repetitions of B_1, $k_2 - k_1$ be the number of repetitions of B_2 and so on $k_\ell - k_{\ell-1}$ be the number of repetitions of B_ℓ. Since we only consider the $|\nu|$ largest b_i's in the definition of W, without loss of generality we assume that $k_\ell = |\nu|$.

Therefore, W can be expressed as

$$W = \sum_{i=1}^{k_1} \beta_i B_1 + \sum_{i=k_1+1}^{k_2} \beta_i B_2 + \cdots + \sum_{i=k_{\ell-1}+1}^{|\nu|} \beta_i B_\ell. \tag{3.15}$$

The key idea to show that $F_\nu(W) \geq 0$ is to introduce a parametric version of the problem. This is done by introducing polynomials $B_i(t)$'s in a variable $t \in [0,1]$ such that for some choice $t^* \in [0,1]$ we recover the original B_i's. If $k = |\nu|$, then showing $F_\nu(W) \geq 0$ is equivalent to showing,

$$W(t)^k \geq h(t), \text{ for } t \in [0,1], \tag{3.16}$$

where

$$h(t) := \sum_{i=1}^{\ell} \sum_{\mu \in K_i} B_i(t)^{k-|\mu|} W(t)^{|\mu|},$$

and the index sets K_i are defined as

$$K_i := \left\{ \mu : |a_\mu|^{1/(|\nu-\mu|)} = B_i \right\}.$$

To prove (3.16), Westerfield chooses $B_i(t)$'s such that $W(t)$ is independent of t and remains equal to W. For t either 0 or 1, we will use the induction hypothesis to obtain that $W^k \geq h(t)$. For $t \in (0,1)$, the key observation will be that the function $h(t)$ is convex in t, and since W^k dominates $h(t)$ at the extremities, it dominates it for all $t \in (0,1)$. We now give the details of the proof for $\ell > 1$.

The polynomials $B_1(t), B_2(t)$ will be linear in t; for $i > 2$, $B_i(t) := B_i$, i.e., they are constants. Let $B_1(t) := \alpha t + \beta$ and $B_2(t) := \gamma t + \delta$. The choice of the four parameters α, β, γ and δ is governed by the following four constraints:

C1. To apply the induction hypothesis at $t = 0$, we ensure that $B_1(0) = B_2(0)$, i.e., $\beta = \delta$.

C2. To apply the induction hypothesis at $t = 1$, we ensure that $B_2(1) = 0$. This

implies that the number of non-zero B_i's is $(\ell - 1)$.

C3. To ensure that W is independent of t, we require that

$$\alpha \sum_{i=1}^{k_1} \beta_i + \gamma \sum_{i=k_1+1}^{k_2} \beta_i = \alpha y_{k_1} + \gamma(y_{k_2} - y_{k_1}) = 0.$$

C4. To ensure that there is a t^* such that $B_1(t^*) = B_1$ and $B_2(t^*) = B_2$ we require that $(B_1 - \beta)/\alpha = (B_2 - \beta)/\gamma$.

Solving for α, β, γ we get the following:

$$\beta = \frac{(B_1 - B_2)y_{k_1} + B_2 y_{k_2}}{y_{k_2}},\ \alpha = \beta \frac{(y_{k_2} - y_{k_1})}{y_{k_1}},$$

and $\gamma = -\beta$. Since $y_{k_2} > y_{k_1}$, β is in the interval (B_2, B_1), $B_1(t)$ is increasing and $B_2(t)$ is decreasing, but both are positive.

The conditions C1 and C2 imply that W^k dominates $h(0)$ and $h(1)$. As mentioned earlier, to prove that W^k dominates $h(t)$ for all $t \in (0,1)$ it suffices to show that $h(t)$ is a convex function for $t \in (0,1)$. This will follow if we show that the second derivative of h is positive. Since only $B_1(t)$ and $B_2(t)$ depend on t, we have

$$h''(t) = \sum_{\mu \in K_1} (k - |\mu|)(k - |\mu| - 1)\alpha^2 B_1(t)^{k-|\mu|-2} W^{|\mu|} + \sum_{\mu \in K_2} (k - |\mu|)(k - |\mu| - 1)\gamma^2 B_2(t)^{k-|\mu|-2} W^{|\mu|}.$$

Since $B_1(t), B_2(t) > 0$, for $t \in (0,1)$ and $\gamma^2 > 0$, it follows that $h''(t) \geq 0$, therefore, $h(t)$ is a convex function as desired. This completes the proof of the induction case.

Q.E.D.

Now, we will prove that the bound in (3.12) is a bound on the positiveness of the non-zero derivatives of F_ν.

LEMMA 14. *The bound $\mathscr{W}(F_\nu)$ in* (3.12) *is a bound on the absolute positiveness of F_ν given in* (3.2).

Proof. Consider a non-zero derivative of F_ν of order λ:

$$\frac{\nu!}{(\nu-\lambda)!}X^{\nu-\lambda}+\sum_{\substack{\mu\prec\nu\\ \lambda\preceq\mu}}\frac{\mu!}{(\mu-\lambda)!}a_\mu X^{\mu-\lambda}, a_\mu<0.$$

Since

$$\frac{\nu!}{(\nu-\lambda)!}>\frac{\mu!}{(\mu-\lambda)!},$$

for all $\mu\prec\nu$, we see that for all non-zero partial derivatives of order λ, $F_\nu^{(\lambda)}(X)$,

$$F_\nu(X)\leq F_\nu^{(\lambda)}(X),$$

for all $X\geq 0$. Hence, from Lemma 13, we conclude that the bound in (3.12) is a bound on the absolute positiveness of F_ν. **Q.E.D.**

REMARK 1. *Some remarks on the proof above.*

1. *The polynomial F_ν in* (3.2) *is monic. But, in general, the leading coefficient of F_ν, $a_\nu>0$, may not be equal to 1. For such polynomials, the definition of b_i is taken to be*

$$b_i := \textit{the ith largest number in } \left(\left|\frac{a_\mu}{a_\nu}\right|^{\frac{1}{|\nu-\mu|}} : 0\preceq\mu\prec\nu\right).$$

The definition of B_i changes analogously as well. Also, the proof of Lemma 13 remains unchanged with these definitions. We will later need special notation for b_1 and b_2, so we define

$$R(F_\nu):=b_1 \text{ and } \rho(F_\nu):=b_2. \tag{3.17}$$

Note that these definitions are analogous to the definitions of $R(f)$ and $\rho(f)$ in (2.3).

2. *A tentative approach to prove Lemma 13 is to show that $\mathscr{W}(F_\nu(X))$ is at least as large as the univariate Westerfield bound $W(F_\nu(x))$ (see Proposition 11) applied to the univariate polynomial $F_\nu(x)$ (defined in* (3.14)*). This appears likely, but we leave this as an open problem. Nevertheless, we do not use $W(F_\nu(x))$ to bound the positiveness of the multivariate polynomial $F_\nu(X)$ because computing it is not as efficient as $\mathscr{W}(F_\nu)$ (this is addressed in Section 3.3) and it does not have an apriori closed form expression like $\mathscr{W}(F_\nu)$ since the coefficients of $F_\nu(x)$ depends on the number of monomials having the same total degree in $F_\nu(X)$.*

We now generalize the bound $\mathscr{W}(F_\nu)$ to arbitrary multivariate polynomials. Consider an arbitrary multivariate polynomial

$$F(X) := \sum_{\mu \in \mu(F) \subseteq \mathbb{N}^n} a_\mu X^\mu + \sum_{\nu \in \nu(F) \subseteq \mathbb{N}^n} a_\nu X^\nu \in \mathbb{R}[X] \tag{3.18}$$

where

$$\mu(F) := \left\{\mu \in \mathbb{N}^n : a_\mu < 0\right\}.$$

$$\nu(F) := \left\{\nu \in \mathbb{N}^n : a_\nu > 0\right\}.$$

We would like to partition F as follows:

$$F = \sum_{\nu \in \nu(F)} F_\nu + G \tag{3.19}$$

where each F_ν is of the form given in (3.2) and G is a polynomial which is a sum of only positive monomials. As in [15], we consider a negative monomial $a_\mu X^\mu$, and assign it to a unique $\nu \in \nu(F)$. [2] This is done by establishing a special function $s : \mu(F) \to \nu(F)$ as

[2] Instead of doing the assignment arbitrarily, we do it in a special manner because this will later help us in reinterpreting Hong's bound.

follows: Given $\mu \in \mu(F)$, consider the set of indices ν' such that

$$\left|\frac{a_\mu}{a_{\nu'}}\right|^{1/|\nu'-\mu|} = \min_{\nu}\left\{\left|\frac{a_\mu}{a_\nu}\right|^{1/|\nu-\mu|} : a_\nu > 0, \quad \mu \prec \nu\right\}.$$

Define $s(\mu)$ to be an arbitrary element of this set; We can now partition F as desired in (3.19): Given $\nu \in \nu(F)$, define

$$F_\nu(X) := a_\nu X^\nu + \sum_{s(\mu)=\nu} a_\mu X^\mu. \tag{3.20}$$

The polynomials F_ν's partition F because the function $s : \mu(F) \to \nu(F)$ is well defined, and hence every negative monomial is associated with a unique positive monomial. For all the $\nu \in \nu(F)$ that are not in the range of the function $s(\cdot)$, F_ν is defined to be the zero polynomial; these monomials appear in G in (3.19). The resulting partition of F is called **a slope partitioning of** F from now onwards. [3]

We are now in a position to define a generalization of the Westerfield's root bound to the multivariate setting. Consider a polynomial F_ν, as in (3.20), in the slope partition of F. From Lemma 13, we know that $\mathscr{W}(F_\nu)$ is a bound on absolute positiveness of F_ν. Taking the maximum of all these bounds we obtain the following bound on the absolute positiveness of F:

$$\mathscr{W}(F) := \max_{\nu \in \nu(F)} \mathscr{W}(F_\nu). \tag{3.21}$$

This is summarized in the following main result of this section:

THEOREM 15. *Given a multivariate polynomial F as defined in* (3.18)*, $\mathscr{W}(F)$ as defined in* (3.21) *is a bound on the absolute positiveness of F.*

From here on, we refer to the bound in (3.21) as the **Westerfield bound on absolute positiveness** in the multivariate setting.

[3] The choice in the definition of the function $s(\mu)$ means that there may be more than one way of partitioning F that yields a slope partition.

3.2.1 Generalizing Lagrange's Real Root Bound to the Multivariate Setting

In this subsection, we will generalize Lagrange's bound in (2.3) to the multivariate setting. We first note that the Westerfield's bound defined in (3.21) has a generic form among the known bounds that use b_i's in their definition. One such well known bound was given by Hong [25]. For the polynomial in (3.18), Hong's bound is

$$H(F) := \frac{1}{\Omega_n} \max_{a_\mu<0} \min_{\substack{a_\nu>0 \\ \mu \prec \nu}} \left| \frac{a_\mu}{a_\nu} \right|^{\frac{1}{|\nu-\mu|}}.$$

Hong showed that $H(F)$ is a bound on the absolute positiveness of F. To the best of our knowledge, we do not know any published bounds that are better than Hong's bound in the multivariate setting. Now, using the partition of F in (3.19), we can reinterpret $H(F)$ as

$$H(F) := \max_{\nu \in \nu(F)} H(F_\nu), \tag{3.22}$$

where $H(F_\nu)$ is defined as,

$$H(F_\nu) := \frac{1}{\Omega_n} \max_{s(\mu)=\nu} \left| \frac{a_\mu}{a_\nu} \right|^{\frac{1}{|\nu-\mu|}} = \frac{R(F_\nu)}{\Omega_n},$$

and $R(F_\nu)$ is as in (3.17).

Before we generalize Lagrange's bound to the multivariate setting, we associate certain linear combinations of b_i's, defined in (3.3), with a polynomial F_ν as in (3.2). Recall that $y_{|\nu|}$ is the positive root of $P_{|\nu|}$. Given a sequence $\overline{\alpha} := (\alpha_1, \ldots, \alpha_k)$ of k numbers such that $\sum_{i=1}^{k} \alpha_i = y_{|\nu|}$, associate the following number with the polynomial F_ν:

$$\mathscr{B}_{\overline{\alpha}}(F_\nu) := \sum_{i=1}^{k} \alpha_i b_i. \tag{3.23}$$

For instance, if α_i's are equal to β_i's as defined in (3.6), then $\mathscr{B}_{\overline{\alpha}}(F_\nu)$ is the Westerfield

bound, $\mathscr{W}(F_\nu)$. Note, however, not all choices of α_i's give us a bound on the absolute positiveness of F_ν. For an arbitrary multivariate polynomial F and a sequence α, we define the number

$$\mathscr{B}_{\overline{\alpha}}(F) := \max_{\nu \in \nu(F)} \mathscr{B}_{\overline{\alpha}}(F_\nu),$$

where F_ν's are the polynomials in the slope partition of F. Note that from the definition of $\mathscr{B}_{\overline{\alpha}}(F_\nu)$ and $R(F_\nu)$ we have

$$\mathscr{B}_{\overline{\alpha}}(F_\nu) \leq R(F_\nu) \sum_i \alpha_i = R(F_\nu) y_{|\nu|}.$$

Since $y_{|\nu|} < 1/\Omega_n$, it follows that $\mathscr{B}_{\overline{\alpha}}(F_\nu) < H(F_\nu)$ and consequently $\mathscr{B}_{\overline{\alpha}}(F) < H(F)$. Therefore, all numbers $\mathscr{B}_{\overline{\alpha}}(F_\nu)$ are smaller than Hong's bound, though not all of them are bounds on absolute positiveness. But it is not clear if $\mathscr{B}_{\overline{\alpha}}(F)$ can be computed as efficiently as $H(F)$. We know that $H(F)$ can be computed in $O(m \log^n m)$ time, where m is the number of non-zero monomials in F [35]. Note that if $\overline{\alpha}$ was just a singleton element, namely $y_{|\nu|}$, then the resulting number is $y_{|\nu|}R(F_\nu)$ is closest to $H(F_\nu)$. Therefore, if we want to attain similar running time as Hong's bound, we should consider sequences $\overline{\alpha}$ that have constantly many terms. With this in mind, we define a spectrum of bounds, with Westerfield's bound on one extreme and Hong's bound at the other extreme, such that there is a tradeoff between the complexity of computing the bound and the tightness of the bound.

Given a polynomial F_ν, define the sequences

$$\overline{\alpha}_j := (\beta_1, \beta_2, \ldots, \beta_j, y_{|\nu|} - (\beta_1 + \cdots + \beta_j))$$

for $j = 0, \ldots, |\nu|$ and β_i's are as defined in (3.6). These sequences yield a hierarchy of bounds for F_ν that satisfy the relation

$$\mathscr{W}(F_\nu) = \mathscr{B}_{\overline{\alpha}_{|\nu|}}(F_\nu) \leq \mathscr{B}_{\overline{\alpha}_{|\nu|-1}}(F_\nu) \leq \cdots \leq \mathscr{B}_{\overline{\alpha}_1}(F_\nu) \leq \mathscr{B}_{\overline{\alpha}_0}(F_\nu).$$

For an arbitrary multivariate polynomial F, as defined in (3.18), we use the slope partition of F and define the **hierarchy of Westerfield-type bounds** as follows:

$$\mathscr{W}(F) = \max_{\nu \in \nu(F)} \mathscr{B}_{\overline{\alpha}_{|\nu|}}(F_\nu) \leq \cdots \leq \max_{\nu \in \nu(F)} \mathscr{B}_{\overline{\alpha}_1}(F_\nu) \leq \max_{\nu \in \nu(F)} \mathscr{B}_{\overline{\alpha}_0}(F_\nu). \tag{3.24}$$

Clearly, all the bounds in the hierarchy are strictly smaller than $H(F)$. But the larger bounds are easier to compute since they require computing fewer β_i's. We will specially be interested in the second largest bound of this hierarchy. More specifically, given a polynomial F_ν as in (3.2), recall the definitions of $R(F_\nu)$ and $\rho(F_\nu)$ from (3.17); since $\beta_1 = n$, we have $\overline{\alpha}_1 = (n, y_{|\nu|} - n)$. The **Lagrange bound on absolute positiveness** of a polynomial F is defined as

$$\mathscr{L}(F) := \max_{\nu \in \nu(F)} \mathscr{L}(F_\nu), \tag{3.25}$$

where

$$\mathscr{L}(F_\nu) := nR(F_\nu) + (y_{|\nu|} - n)\rho(F_\nu) \tag{3.26}$$

and F_ν are the polynomials in the slope partition of F.

REMARK 2. *For a monic univariate polynomial* $f(x) := x^d + \sum_{i=0}^{d-1} a_i x^i$, $a_i \in \mathbb{R}$, *the bound in* (3.25) *is* $R(f) + (y_d - 1)\rho(f)$, *where* $y_d < 2$. *Hence, it is always better than Lagrange's bound of* f *in* (2.3). *Now, comparing the bound in* (3.25) *for* f *with the improvement given in [6, Thm. 3.1], we see that when* $R(f)$ *is equal to* $\rho(f)$, *the bound given in* (3.25) *for* f *is better than the improvement in [6, Thm. 3.1]. In other cases, it is not clear if the bound for* f *given in* (3.25) *improves upon the bound for* f *in [6, Thm. 3.1].*

We will now quantify the improvement of Westerfield-type bounds over Hong's bound

LEMMA 16. *For an arbitrary polynomial* F *of total degree* d *as in* (3.18), *Hong's bound*

and any Westerfield-type bound satisfy the following inequality:

$$H(F) - \mathscr{B}_{\overline{\alpha}}(F) > c_{n,d} \frac{\Omega_n^d}{d(1-\Omega_n e^{-1/d})} \cdot \left|\frac{a}{b}\right|^{1/d},$$

where a and b are the smallest and largest coefficients of F in absolute value, respectively, and $c_{n,d}$ is as defined in (3.7)*.*

Proof. Consider the special polynomial F_ν in the slope partition of F where $\mathscr{B}_{\overline{\alpha}}(F)$ is achieved, i.e, $\mathscr{B}_{\overline{\alpha}}(F) = \mathscr{B}_{\overline{\alpha}}(F_\nu)$. Assuming $k = |\nu|$, we have $\mathscr{B}_{\overline{\alpha}}(F_\nu) \leq y_k R(F_\nu)$. Moreover, by definition of Hong's bound (3.22), we know that $H(F) \geq H(F_\nu)$. So,

$$H(F) - \mathscr{B}_{\overline{\alpha}}(F) \geq H(F_\nu) - \mathscr{B}_{\overline{\alpha}}(F_\nu) \geq \left(\frac{1}{\Omega_n} - y_k\right) R(F_\nu).$$

As $k \leq d$, and since y_k monotonically converge to $1/\Omega_n$, we obtain that

$$H(F) - \mathscr{B}_{\overline{\alpha}}(F) \geq \left(\frac{1}{\Omega_n} - y_d\right) R(F_\nu).$$

By applying the lower bound on $(1/\Omega_n - y_d)$ from Lemma 12, we get

$$H(F) - \mathscr{B}_{\overline{\alpha}}(F) > c_{n,d} \frac{\Omega_n^d}{d(1-\Omega_n e^{-1/d})} R(F_\nu).$$

To lower bound $R(F_\nu)$, observe that $|\nu - \mu| \leq d$, for all $\nu, \mu \in \nu(F) \cup \mu(F)$; therefore, $R(F_\nu) \geq (a/b)^{1/d}$, for all $\nu \in \nu(F)$, where a and b are as defined in the statement of the lemma. This yields us the claimed result. **Q.E.D.**

In the next section, we will give an algorithm that computes $\mathscr{L}(F)$ and runs in time $O(m \log^n m)$.

3.3 Algorithm for the Generalized Lagrange bound

Given a multivariate polynomial F as defined in (3.18) with m monomials, we need an efficient sub-routine to compute the set $\{\nu : \nu \in \nu(F), \mu \prec \nu\}$, for a given $\mu \in \mu(F)$. This is a special case of orthogonal range searching in computational geometry. We give a brief description of a data structure to solve this problem; more details can be found in [10, Chapter 5].

3.3.1 Range Trees

Consider the following problem: Given a set of m input real numbers, and a number $q \in \mathbb{R}$, find the subset of the input that is larger than q. This problem is called a range query in one dimension. A range query is more generic in the sense that one can input an interval $[a,b]$ and ask for all the numbers in the given input that are in the range $[a,b]$. In this paper though, we restrict ourselves to half-open range queries. A 1**-dimensional range tree** is simply a balanced binary search tree (BST) on the input set of m numbers. The input numbers are stored at the leaves of the tree in an ascending order. The numbers stored at the internal nodes come from the input set and guide the search on a query q. Starting with the root we proceed recursively as follows: If the value at a node u is less than or equal to q we go to the right subtree of u; otherwise, we go to the left subtree of u; on reaching a leaf, we output all numbers associated with the leaves to the right of this leaf. To describe the output we need the notion of **the canonical subset of a node** u, i.e,. the set of points stored at the leaves of the sub-tree rooted at u. Since there is a canonical set associated with each node, the number of canonical sets in the output to a query is $O(\log m)$. We index them appropriately, and the output to a range query is an appropriate subset of these indices, namely, on the search path whenever we go left at a node u, we include the index of the canonical subset of its right child in the output. This approach naturally generalizes to higher dimensions via the following recursive definition of an n-dimensional range tree:

1. The first level tree is a balanced binary search tree on the x_1-coordinates of the input points in $\mathbb{R}^n$.

2. For every node u in the first level tree, an associated $(n-1)$-dimensional tree is constructed on the remaining $(n-1)$-coordinates of the points in the canonical subset of u. This tree is referred to as the **associated tree** of u(see Figure 3.1 for illustration).

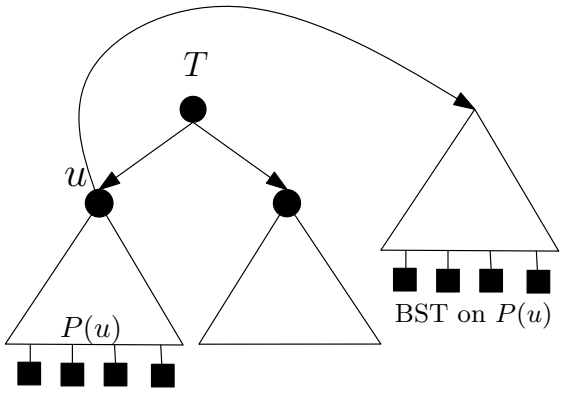

Figure 3.1: A 2-dimensional range tree.

To compute $\mathscr{L}(F)$, we construct the range tree $T(F)$ on the set $v(F)$. Let $W_1,\ldots,W_r$ be the canonical subsets of the nth level trees of $T(F)$. It is known that [10, Chap. 5, p. 105],

$$\sum_{k=1}^{r} |W_k| = O(m\log^n m). \tag{3.27}$$

A query to $T(F)$ is a vector of the form $\mu \in \{0,\ldots,\delta\}^n$, where δ is the maximum degree of any x_i in F, $i \in [n]$. The output of the query is the set

$$O(\mu) := \{k \in [r] : W_k \text{ is a canonical subset in the output to query } \mu\},$$

i.e., the set of indices of at most $O(\log^n m)$ many W_k's whose disjoint union forms the dominating set. The tree $T(F)$ can be constructed in $O(m\log^{n-1} m)$ algebraic operations [10, p. 110, Thm 5.9].

3.3.2 Algorithm

We begin with some definitions that will be useful in the description of the algorithm. For every non-zero monomial $a_\lambda X^\lambda$ in F, define the point

$$p_\lambda := (|\lambda|, \log(1/|a_\lambda|)) \in \mathbb{R}^2.$$

For a given $k \in O(\mu)$, let

$$\sigma_{\mu,k} := \log \min_{\nu \in W_k} \left| \frac{a_\mu}{a_\nu} \right|^{1/|\nu-\mu|} = \min_{\nu \in W_k} \frac{\log |a_\mu| - \log a_\nu}{|\nu - \mu|}. \tag{3.28}$$

Geometrically, $(\log |a_\mu| - \log a_\nu)/|\nu - \mu|$ is the slope of the line passing through the points p_μ and p_ν. It is not hard to see that $\sigma_{\mu,k}$ is the slope of the lower tangent from the point p_μ to the lower convex hull of the points p_ν, for all $\nu \in W_k$. Also, $s(\mu)$ defined in Section 3.2 satisfies the following property:

$$\min_{k \in O(\mu)} \sigma_{\mu,k} = \left| \frac{a_\mu}{a_{s(\mu)}} \right|^{1/|s(\mu)-\mu|}.$$

We first describe a suboptimal algorithm that will compute $\mathscr{L}(F)$ in $O(m \log^{n+1} m)$ algebraic operations. It is obtained by a slight modification to one of the algorithms of Mehlhorn and Ray [35, p. 7]. Given F, we construct $T(F)$. We query $T(F)$ on every $\mu \in \mu(F)$ to compute $O(\mu)$. Now for a given pair (μ, k), for $k \in O(\mu)$, we compute the value of $\sigma_{\mu,k}$ by computing the lower tangent of the point p_μ to the lower hull of points $\{p_\nu : \nu \in W_k\}$. Then we compute the $s(\mu)$, for each μ in $\mu(F)$, from which we compute the required partition of F, and thereby the bound $\mathscr{L}(F)$. Since the cost of constructing a lower hull for a given W_k is $O(|W_k| \log |W_k|)$, from (3.27) we see that the total cost of hull construction is $O(m \log^{n+1} m)$. So the algorithm runs in $O(m \log^{n+1} m)$ algebraic operations. We now improve the running time of this algorithm to $O(m \log^n m)$. Similar to Mehlhorn and Ray, we construct an $(n-1)$-dimensional range tree $T'(F)$ on the last

$(n-1)$ coordinates of the points in $\nu(F)$ and then process the monomials in the decreasing order of the remaining x_1-coordinate. Let $W'_1,\dots,W'_r$ be the canonical subsets of $T'(F)$. Define

$$W'_{\mu,k} := \left\{\nu \in W'_k : \mu \prec \nu, \text{ i.e., } \nu \text{ dominates } \mu \text{ in } x_1 \text{ and } (x_2,\dots,x_n)\right\},$$

Now we redefine $\sigma_{\mu,k}$ as follows:

$$\sigma_{\mu,k} := \log \min_{\nu \in W'_{\mu,k}} \left|\frac{a_\mu}{a_\nu}\right|^{1/|\nu-\mu|}.$$

The algorithm is the following:

Algorithm: `LagrangeBound`

INPUT: A multivariate polynomial $F(X)$ as in (3.18).

OUTPUT: The bound $\mathscr{L}(F)$ as in (3.25).

1. Construct the $(n-1)$-dimensional range tree $\mathscr{T}'(F)$.
2. Sort the entries in the arrays $W'_1, \ldots, W'_r$ by their x_1-coordinate.
3. For each $k \in [r]$, let j_k be the size of W'_k.
4. $H_k \leftarrow \emptyset$.

   ◁ *Lower hull corresponding to the points p_ν such that*

   ◁ *$\nu \in W'_{\mu,k}$; data structure for H_k is as in [12] where with each*

   ◁ *point on the hull we store a representative tuple ν.*

5. **For all** $\mu \in \mu(F)$ in decreasing order of x_1-coordinate **do**:

   Query $\mathscr{T}'(F)$ to compute $\mathrm{Out}(\mu)$.

5.a. **For all** $k \in \mathrm{Out}(\mu)$ **do**:

   $D_{\mu,k} \leftarrow \emptyset$.

5.b. **For all** $\nu \in W'_k$ starting from the index j_k **do**:

5.c. **If** ν dominates μ in the x_1-coordinate **then**

   $D_{\mu,k} \leftarrow \nu \cup D_{\mu,k}$; $j_k \leftarrow j_k - 1$.

5.d. $H_k \leftarrow$ the lower hull of the points $H_k \cup \{p_\nu, \nu \in D_{\mu,k}\}$.

5.e. Compute $\sigma_{\mu,k}$, the slope of the lower tangent from p_μ to H_k.

5.f. Let $s(\mu)$ be an index in $\cup_{k \in \mathrm{Out}(\mu)} W'_{\mu,k}$ attaining $\min_{k \in \mathrm{Out}(\mu)} \sigma_{\mu,k}$.

6. Compute the partitioning of F as in (3.19) using the values $s(\mu)$.
7. Compute $\mathscr{L}(F_\nu)$ as in (3.26), for each $\nu \in \nu(F)$.
8. Output $\max_{\nu \in \nu(F)} \mathscr{L}(F_\nu)$.

Since the lower hulls H_k are updated dynamically, we use the data structure for dynamic convex hull given by Jacob and Brodal [12] to store points in H_k. We analyse the algorithm in the real RAM model where the usual arithmetic operations and operations such as exponentiation are charged unit cost. The main result of this section is the following:

THEOREM 17. *The number of algebraic operations required by the algorithm above is* $O(m\log^n m)$.

Proof. The running time of the algorithm can be analyzed as follows:

1. The cost of constructing the range tree $T'(F)$ is $O(m\log^{n-1} m)$.

2. Sorting the canonical subsets by their x_1-coordinates takes $O(m\log^{n-1} m)$ algebraic operations. This is because the canonical subsets of an $(n-1)$st level tree with m leaves can be sorted in time $O(m\log m)$ time using merge sort [16].

3. The cost analysis of processing the negative monomials can be broken down as follows:

   (a) The cost of scanning a given W'_k in the for-loop at Step 5.b. to compute $D_{\mu,k}$'s over all iterations is $|W'_k|$. This is because we are processing the negative monomials in the decreasing order of their x_1-coordinates. So if a W'_k is in the output of μ, we can start scanning W'_k from the index where the previous scan of W'_k had stopped; this is done by the index j_k. So the total cost of this step over all iterations is $\sum_k^r |W'_k| = O(m\log^{n-1} m)$.

   (b) The total cost of updating the hull H_k corresponding to a given W'_k is $O(|W'_k|\log|W'_k|)$. This is because, if a W'_k is in the output of two points μ and μ' such that $x_1 \geq x'_1$, then $W'_{\mu,k} \subseteq W'_{\mu',k}$. Therefore, while processing μ', H_k only needs to be updated with points p_ν such that $\nu \in (W'_{\mu',k} \setminus W'_{\mu,k})$. Now the amortized cost of each update, which could be an insertion or a deletion, can be done in $O(\log|W'_k|)$ operations [12, Thm 1]. So the total cost of constructing lower hulls H_k over all iterations is
   $$\sum_{k=1}^{r} |W'_k|\log|W'_k| = O(m\log^n m).$$

   (c) The cost of computing the lower tangent of a point p_μ to the hull H_k is $O(\log|W'_k|)$. Since there can be $O(\log^{n-1} m)$ many W'_k's in the output of μ, the

cost of computing the lower tangents of p_μ is $O(\log^n m)$. So the total cost of this step over m iterations is $O(m \log^n m)$.

(d) Computing $s(\mu)$ takes $O(\log^{n-1} m)$ operations for a given μ. Hence the total cost of this step is $O(m \log^{n-1} m)$.

4. The cost of computing the required partition of F using $s(\mu)$, for all $\mu \in \mu(F)$, is $O(m)$.

5. Since the bound $\mathscr{L}(F)$ is the maximum over the Lagrange bound of the F_ν's in the partition, it can be computed in $O(m)$ time.

6. Computation of $L(F_\nu)$, for each ν in $\nu(F)$, takes $O(m)$ time.

From the steps above, we see that the total cost of the algorithm is bounded by $O(m \log^n m)$.

Q.E.D.

REMARK 3. *A few remarks on the algorithm above.*

1. *The space complexity of the algorithm* `LagrangeBound` *is as follows: Construction of range tree requires $O(m \log^{n-2} m)$ space [10, p. 110, Thm 5.9]. The space complexity of the lower hulls constructed is $O(m \log^{n-1} m)$. This is because the total number of points over all the lower hulls is $O(m \log^{n-1} m)$ and the number of edges in a convex hull is linear in the number of points. Since any output to a query is of size at most m, the space required for writing down the querying output is $O(m)$. Hence, the asymptotic space complexity of* `LagrangeBound` *is $O(m \log^{n-1} m)$.*

2. *Computation of $L(F_\nu)$ in the penultimate step of the algorithm needs the positive root, $y_{|\nu|}$, of the canonical polynomial $P_{|\nu|}$. But $y_{|\nu|}$ needs to computed only once and can be stored for future iterations of the algorithm. Here we note that an ε approximation to $y_{|\nu|}$ can be computed with $O(|\nu| \log(1/\varepsilon))$ Newton iterations [32, Lemma. 2.2]. It must be noted that by modifying the routine* `LagrangeBound`*, any arbitrary Westerfield-type*

bound that considers constant many b_i's can be computed in asymptotically same running time as `LagrangeBound`*. Also,* `LagrangeBound` *can be modified to compute Westerfield's bound defined in* (3.21)*. But, in order to compute the bound in* (3.21)*, we need to compute $y_1, \ldots, y_{|\nu|-1}$ along with $y_{|\nu|}$. This means that the number of Newton iterations needed to compute the y_k's will be $O(|\nu|^2 \log(1/\varepsilon))$.*

3. *In Remark 1, we had mentioned that one can apply univariate Westerfield bound $W(F_\nu)$ to the polynomial $F_\nu(x)$, instead of $W(F_\nu)$, to compute a bound on the positiveness of $F_\nu(X)$. What is the cost of computing the univariate polynomials $F_\nu(x)$ given in* (3.14)*? The degree of every monomial and the subsequent addition of the coefficients of the monomials of the same degree needs extra work. Computing the total degree of all the monomials needs $\Theta(mn)$ operations. Now, adding the monomials of the same total degree in every $F_\nu(X)$ can be done as follows: sort the monomials by their total degree, which takes $O(m \log m)$ time, and add the coefficients of the monomials with the same total degree; or, if we are allowed to use extra space, we can bucket sort the monomials using $O(|\nu|)$ space and add the coefficients of the monomials in the same bucket in $O(m)$ time. In either case, it is not clear if the extra overhead involved is compensated by an improvement in the quality of the bound.*

To the best of our knowledge, the algorithm of Mehlhorn and Ray for computing Hong's bound [35, p. 7-8], is quite different from `LagrangeBound`. The main differences are as follows:

1. The construction of the range tree in their algorithm seems to be done in an incremental manner, which is not the case in our algorithm.

2. There is no sorting of the canonical subsets by the x_1-coordinate in their algorithm.

3.3.3 Efficiency of the Algorithm

In this section, we address the efficiency of the algorithm `LagrangeBound` for computing Lagrange's bound. For a polynomial in n variables with m monomials, Theorem 17 shows that the number of algebraic operations needed by `LagrangeBound` is $O(m\log^n m)$. This is a consequence of steps $3.(b)$ and $3.(c)$ in the proof of Theorem 17. For a given n-variate polynomial, the algorithm does range querying on m points in $(n-1)$-dimensions. Although the algorithms uses orthogonal range trees for range querying, in principle, it can work with any data structure for range querying. Is it possible that a different data structure for range querying can be used to improve the running time of the algorithm? By applying the lower bound on range querying due to Fredman [22], we can infer that the orthogonal range tree is, in a certain sense, asymptotically the best data structure for range querying.

Let D be any data structure that answers range queries on m points in $(n-1)$-dimensions. Without loss of generality, we assume that there are m possible outputs. Suppose D stores $w_1,\ldots,w_r$ subsets of these m points such that every output to a (half-open) range query is given as a union of at most ℓ pairwise disjoint sets from $w_1,\ldots,w_r$. Then, the cost of the steps $3.(b)$ and $3.(c)$ will be $\Theta((\log m)\sum_{i=1}^{r}|w_k|)$ and $\Theta(m\ell\log m)$, respectively. So, in order to improve the running time of the algorithm, D must have $\sum_{i=1}^{r}|w_k| = o(m\log^{n-1} m)$ and $\ell = o(\log^{n-1} m)$, i.e., either the total size of the sets $w_1,\ldots,w_r$ is small or the output of every query can be expressed as a union of small number of sets from $w_1,\ldots,w_r$; if not, then at least one of the steps among $3.(b)$ or $3.(c)$ will be $\Theta(m\log^n m)$. There is an inherent tradeoff between the **the storage size** $\sum_k |w_k|$ and **the worst case output size** ℓ. To understand this tradeoff, let us consider the case of one-dimensional range querying, i.e., when $n=1$. In this case, there are m possible distinct outcomes of the range queries. Two natural choices for the sets w_k's are either the singleton elements, or the m sets that can possibly appear in the output. In the first case, $\sum_{i=1}^{m}|w_k| = m$ and $\ell = m$; in the second case, $\sum_{i=1}^{m}|w_k| = \Theta(m^2)$ and $\ell = 1$. Therefore, in these two extreme scenarios $\ell\sum_i |w_k| = \Theta(m^2)$. Can this be improved? Not surpris-

ingly, if we consider the canonical sets w_k stored by the balanced binary search tree in one-dimension, then the quantity $\ell \sum_i |w_k| = \Theta(m(\log m)^2)$. But is this bound tight, i.e., do all data structures D satisfy the same inequality?

Fredman [22] answered a related question by showing the following weaker lower bound on range querying:

PROPOSITION 18. *Let D be a data structure that answers range queries on m points*[4] *in $\mathbb{R}^n$ by storing $w_1, \ldots, w_r$ subsets of these m points and using at most ℓ of these sets in answering any range query. Then D must satisfy the following bound*

$$\sum_{k=1}^{r} |w_k| + m\ell = \Omega(m \log^n m).$$

To keep our exposition concise, we refer to [34, p. 69] for the proof of the proposition above. It must be noted that the lower bound above is shown by considering half-open range queries like the ones considered in this paper. So, the lower bound above is applicable to our setting. Note that the algorithm `LagrangeBound` does range querying in $(n-1)$-dimensions for an n-variate polynomial. So, by applying Proposition 18 for $(n-1)$ dimensions, we can infer that the combined complexity of steps 3.(b) and 3.(c) is

$$\Theta(\log m(\sum_{k=1}^{r} |w_k| + m\ell)) = \Theta(m \log^n m).$$

This implies that we cannot improve the running time of the algorithm for computing Lagrange's bound by using a different data structure for range querying.

3.4 Further Directions

In this chapter, we have given generalizations of Westerfield and Lagrange's bound to the multivariate setting. These bounds are quantitative improvements over Hong's bound,

[4] Fredman derives the lower bound by fixing the input to be the m points in the n-dimensional grid.

which was the best known bound so far. We also give an $O(m\log^n m)$ algorithm to compute the Lagrange bound. The asymptotic running time of this algorithm is the same as the asymptotic running time of the best known algorithm [35] for computing Hong's bound. Since the upper bounds of both the algorithms are the same, we ask the following question: Is there a lower bound of $\Omega(m\log^n m)$ on these algorithms in the algebraic decision tree model? An interesting case is when $n = 1$ where we have an $O(m)$ time algorithm for computing Hong's bound [30] and an $\Omega(m\log m)$ lower bound on the computation of Lagrange's bound [39]. However, it must be noted that the linear time algorithm for computing Hong's bound assumes that the monomials can be read in the decreasing order of their degree. More precisely, the question when $n = 1$ would be, without the assumption on the ordering of monomials, can we show a lower bound of $\Omega(m\log m)$ on the computation of Hong's bound?

Another important question in this line of work is to give a bound only on the positiveness of the polynomial. The known bounds in the literature are all bounds on the absolute positiveness of the polynomial. In the univariate setting, this question amounts to bounding the largest positive root of the given polynomial, if one exists. Any algorithm that computes such a root bound must detect if the given polynomial has a positive root or not? It is clear that by using algorithms for root isolation or root approximation, we can detect if the given polynomial has a positive root or not. But, it must be noticed that the running time of the known algorithms for root isolation and root approximation are quadratic in the degree of the polynomial. Hence the challenge is to give an algorithm that detects if the given polynomial has a positive root or not and runs in sub-quadratic time.

Also, the tradeoff in Proposition 18 gives us a lower bound on the arithmetic mean of the total size of the canonical sets and the total number of canonical sets needed for covering query ranges. But in practice, data structures such as range trees need $\Theta(m\log^n m)$ many canonical sets for orthogonal range querying on m points and the sum total size of these sets is $\Theta(m\log^n m)$ as well. In view of this fact, it is more relevant to consider the

geometric mean of the total size of the canonical sets and the number of canonical sets needed for covering all the query ranges. In this respect we make the following stronger claim:

CLAIM 1. *Let D, $w_1, \ldots, w_r$ and ℓ be as in Proposition 18. Then D must satisfy*

$$m\ell \sum_{k=1}^{r} |w_k| = \Omega(m^2 \log^{2n} m).$$

From the AM-GM inequality it follows that the claim above implies Proposition 18. In the next chapter, we will focus on proving the claim above and also show that the balanced binary search tree is near optimal in a more relaxed framework for data structures supporting range querying in one dimension.

Chapter 4

Stronger Tradeoffs for Orthogonal Range Querying in the Semigroup Model

In this chapter, we will strengthen a lower bound on the tradeoffs for orthogonal range querying due to Fredman [22]. Before we state Fredman's lower bound precisely, we set up some notations and definitions: Let X be the set of m points $X_1, \ldots, X_m$ in the n-dimensional grid, i.e.,

$$X := \left[1, 2, \ldots, m^{1/n}\right]^n.$$

The n coordinates of the point X_i are represented as X_{ij}, $j = 1, \ldots, n$. A **one-sided orthogonal range query** on the set X takes as input a $Y \in \mathbb{R}^n$ and outputs

$$R_Y := \{X_i \in X : X_i \leq Y\},$$

where $X_i \leq Y$ iff $X_{ij} \leq Y_j$, for all $j \in [n]$. In this chapter, range queries will always be one-sided. Corresponding to m points in X, we have m range queries whose outputs are $R_j := R_{X_j}$, for $j \in [m]$. A set $D := \{W_1, \ldots, W_r\}$ of subsets of X is a **data structure for**

answering range queries on X if every output to a range query on X is represented as a *disjoint union* over the canonical sets $W_1, \ldots, W_r$. Let $\langle R_j \rangle_D$ denote the set of indices of W_k's used in the representation of R_j. From Chapter 3, recalling the result in Proposition 18,

PROPOSITION 19. *If D is a data structure that answers range queries on X then*

$$\sum_{k=1}^{r} |W_k| + \sum_{j=1}^{m} |\langle R_j \rangle_D| = \Omega(m \log^n m).$$

The result above is a more precise formulation of Proposition 18 as the parameter "$m\ell$" in Proposition 18 is lower bounded by $\sum_{j=1}^{m} |\langle R_j \rangle_D|$. This tradeoff in the lower bound argument in [22] is the central theme of this chapter. Many more lower bounds on orthogonal range querying in different computation models are also known (see [1, p. 7] and [2, p. 11]). To the best of our knowledge, most of these lower bounds are space-time tradeoffs required by a data structure in answering a range query.

Our main results in this chapter are:

1. Motivated by the Claim 1 at the end of Chapter 3, we prove a stronger version of Proposition 19:

   THEOREM 20. *If D is a data structure that answers range queries on X then*

   $$\left(\sum_{k=1}^{r} |W_k| \right) \left(\sum_{j=1}^{m} |\langle R_j \rangle_D| \right) = \Omega(m^2 \log^{2n} m).$$

   Notice that the theorem above is stronger than Claim 1 as the parameter "$m\ell$" in Claim 1 is lower bounded by $\left(\sum_{j=1}^{m} |\langle R_j \rangle_D| \right)$. From the AM-GM inequality it is clear that Theorem 20 implies Proposition 19. Theorem 20 also implies that any data structure $\mathscr{D}$ that is *tight* with respect to Proposition 19, i.e.,

   $$\sum_{k=1}^{r} |W_k| + \sum_{j=1}^{m} |\langle R_j \rangle_D| = \Theta(m \log^n m), \tag{4.1}$$

must satisfy:

$$\sum_{k=1}^{r} |W_k| = \Theta(m \log^n m) \text{ and } \sum_{j=1}^{m} |\langle R_j \rangle_D| = \Theta(m \log^n m). \tag{4.2}$$

The proof of Theorem 20 will be given in Section 4.1.

2. From (4.2), we see that the balanced binary search tree is an optimal data structure in the *boolean* setting where the outputs are represented as disjoint unions over canonical sets. This leaves open the possibility of existence of a more efficient data structure that does not take disjoint unions of its canonical subsets but takes their weighted sum in order to represent an output. In such a relaxed setting, Proposition 19 and Theorem 20 are not applicable. Can balanced binary search tree be an optimal data structure even in this setting? In Section 4.2, we give a positive answer to this question.

   In order to account for data structures which take weighted sums of their canonical sets, we will reinterpret range querying differently from Proposition 19. In the proof of Proposition 19 [34, p. 69, Lemma 9 and 10], the problem of range querying is interpreted in a graph theoretic setting, namely expressing a bipartite graph as a "product" of two bipartite graphs. This can also be interpreted in terms of matrices [13, Sec. 2.2]. Let $U_{m\times r}$ be the incidence matrix of the set X with the canonical sets W_k's, i.e, $U_{ik} = 1$ iff $X_i \in W_k$. Similarly, define $V_{r\times m}$ to be the incidence matrix of the canonical sets W_k's and the outputs R_j's. Let $R_{m\times m}$ be the matrix whose columns are the characteristic vectors of the sets R_j's. To give a proof of Proposition 19, it suffices to derive a lower bound on the optimal value of the following optimization problem:

$$\min \left(||U||_F^2 + ||V||_F^2 \right) \text{ subject to } UV = R, \tag{4.3}$$

   where $R \in \{0,1\}^{m\times m}$, $U \in \{0,1\}^{m\times r}$ and $V \in \{0,1\}^{r\times m}$ and $||.||_F$ refers to the

Frobenius norms of the respective matrices.

In this optimization based formulation of the problem, the objective function aims to minimize the sum of the two parameters we are interested in: The total size of the canonical sets, $||U||_F^2$ and the total number of canonical sets needed to cover all the query ranges, $||V||_F^2$. Every data structure that supports range querying in one dimension is a feasible solution to the problem above. When the entries of the matrices are restricted to be boolean, Proposition 19 implies that the optimal value of the objective function is $\Omega(m \log m)$. Hence, from (4.2), we see that for an optimal solution $(U_{\text{bool}}, V_{\text{bool}})$,

$$||U_{\text{bool}}||_F^2 = \Theta(m \log m) \text{ and } ||V_{\text{bool}}||_F^2 = \Theta(m \log m). \tag{4.4}$$

To extend the bounds in (4.4) for data structures which take weighted sums of their canonical sets, we consider the relaxation of the problem in (4.3) where the matrix entries are allowed to be arbitrary reals. For an optimal solution (U^*, V^*) of this relaxation, we show that

$$||U^*||_F^2 = \Omega(m \log m) \text{ and } ||V^*||_F^2 = \Omega(m \log m).$$

The lower bounds above imply that the balanced binary search tree is near optimal **not only** in the boolean framework but also in a more relaxed setting.

The main idea in proving the lower bounds above is to use the Lagrangian dual of the relaxation and show that

$$||U^*||_F^2 = \text{Trace}((R^t R)^{1/2}) \text{ and } ||V^*||_F^2 = \text{Trace}((R^t R)^{1/2}), \tag{4.5}$$

where we take the principal square-root of a matrix [24]. The result in (4.5) holds

for an **arbitrary** matrix R (see Theorem 21) and is the key technical ingredient in our proof. Then, by taking R to be the matrix corresponding to range querying in one dimension and by using some well established results on explicit forms for the eigenvalues of tri-diagonal matrix (in this case $(R^t R)^{-1}$), we show that

$$\text{Trace}((R^t R)^{1/2}) = \Omega(m \log m).$$

We believe that our proof gives more understanding on the optimality of this bound by relating it to some intrinsic parameters of the matrix R, which is a representation of the range query problem in one-dimension. To the best of our knowledge, our proof is more general than the existing proofs in the literature; e.g., [46] works only in the boolean setting. Whether our proof technique can be generalized to obtain an alternative proof of Proposition 19 in all dimensions remains an open and interesting question.

4.1 Tradeoff between Sizes of Canonical Sets and Outputs to Query Ranges

In this section, we will prove Theorem 20. The proof of the theorem follows by altering the argument in the proof of Proposition 19. Before we proceed, we first present some high level details of the proof of Proposition 19. For the complete details, we refer to [34, p. 69].

The argument relies on interpreting range querying in a graph theoretic setting: Consider the weighted bipartite graph $G(X \cup R, E)$, where $R := \{R_1, R_2, \ldots, R_m\}$ and $E := \{(X_i, R_j) : X_i \in R_j\}$; see Figure 4.1(a) for illustration. The edge $(X_i, R_j) \in E$ is

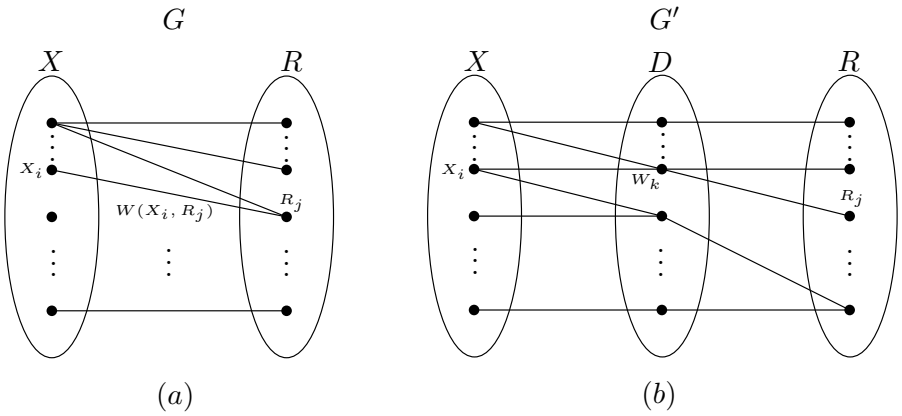

Figure 4.1: (a): Bipartite graph G with the vertex sets X, R and the edge set E. (b): Tripartite graph G' with vertex sets X, D and R.

assigned the weight

$$W(X_i,R_j) := \frac{1}{\prod_{\kappa=1}^{n} (X_{j\kappa} - X_{i\kappa} + 1)}. \tag{4.6}$$

The graph G can be "factored" into a tripartite graph G' whose vertex set is $\{X \cup D \cup R\}$. There is an edge (X_i, W_k) iff $X_i \in W_k$ and there is an edge (W_k, R_j) iff $k \in \langle R_j \rangle_D$; see Figure 4.1(b) for an illustration. Note that the edges of G are a disjoint union over the sets $\{(X_i,R_j) : X_i \in W_k, k \in \langle R_j \rangle_D\}$, for all $k \in [r]$, as every R_j is a disjoint union of W_k's. For every set W_k, define

$$I_k := \{X_i \in X : X_i \in W_k\}$$

i.e., the edges of G' incident on W_k from the left and

$$O_k := \{R_j \in R : k \in \langle R_j \rangle_D\},$$

i.e., the edges of G' incident on W_k from the right. Therefore,

$$|I_k| = |W_k| \text{ and } \sum_{j=1}^{m} |\langle R_j \rangle_D| = \sum_{k=1}^{r} |O_k|.$$

At a high level, the proof of Proposition 19 can be broken down into two steps [34,

p. 69, Lemma 9 and p. 71, Lemma 10]:

Step 1. For every $k \in [r]$,

$$\sum_{\substack{X_i \in I_k \\ R_j \in O_k}} W(X_i, R_j) = \sum_{\substack{X_i \in I_k \\ R_j \in O_k}} \frac{1}{\prod_{\kappa=1}^{n} (X_{j\kappa} - X_{i\kappa} + 1)} \leq (2\pi)^n (|I_k| + |O_k|), \tag{4.7}$$

where n is the dimension of the points in X. The outline of the proof is as follows. Let $M_j = \max \{X_{ij} : X_i \in I_k\}$. Define the **associate sets** of W_k as

$$B := \{(M_1 - X_{i1}, M_2 - X_{i2}, \ldots, M_n - X_{in}) : X_i \in I_k\}$$

and

$$C := \{(X_{j1} - M_1, X_{j2} - M_2, \ldots, X_{jn} - M_n) : R_j \in O_k\}.$$

Since every term of the form

$$(X_{j\kappa} - X_{i\kappa} + 1) = (M_\kappa - X_{i\kappa} + X_{j\kappa} - M_\kappa + 1),$$

the summation in (4.7) is equal to

$$\sum_{\substack{X_i \in I_k \\ R_j \in O_k}} \frac{1}{\prod_{\kappa=1}^{n} (X_{j\kappa} - X_{i\kappa} + 1)} = \sum_{\substack{u \in B \\ v \in C}} \frac{1}{\prod_{\kappa=1}^{n} (u_\kappa + v_\kappa + 1)}.$$

Then, by applying a generalized version of Hilbert's inequality for points with natural numbers as their coordinates, one obtains

$$\sum_{\substack{u \in B \\ v \in C}} \frac{1}{\prod_{\kappa=1}^{n} (u_\kappa + v_\kappa + 1)} \leq (2\pi)^n (|B| + |C|) = (2\pi)^n (|I_k| + |O_k|). \tag{4.8}$$

Step 2. The second step is to show that the total sum of weights over all edges in E satisfies

$$\sum_{(X_i,R_j)\in E} W(X_i,R_j) = \sum_{k=1}^{r} \sum_{\substack{X_i\in I_k \\ R_j\in O_k}} W(X_i,R_j) = \Omega(m\log^n m). \tag{4.9}$$

By summing the inequality in (4.7) over all W_k and applying the lower bound from (4.9), we get the claim in Proposition 19.

We now give the proof of Theorem 20. *Proof.* We consider the same set X as in Fredman's proof, i.e.,

$$X := \left[1,2,\ldots,\lfloor m\rfloor^{1/n}\right]^n.$$

Since $\sum_{k=1}^{r}|W_k| = \sum_{k=1}^{r}|I_k|$ and $\sum_{j=1}^{m}|\langle R_j\rangle_D| = \sum_{k=1}^{r}|O_k|$, we will show that

$$\sum_{k=1}^{r}|I_k|\cdot\sum_{k=1}^{r}|O_k| = \Omega(m^2\log^{2n} m).$$

To prove this, consider a pair of canonical sets, W_k and W_ℓ. Using the same weight function as in (4.6) on the edges of G, we have

$$\sum_{\substack{X_i\in I_k \\ R_j\in O_k}}\sum_{\substack{X_c\in I_\ell \\ R_d\in O_\ell}} W(X_i,R_j)\cdot W(X_c,R_d) = \sum_{\substack{X_i\in I_k \\ R_j\in O_k}}\sum_{\substack{X_c\in I_\ell \\ R_d\in O_\ell}} \frac{1}{\prod_{\kappa=1}^{n}(X_{j\kappa}-X_{i\kappa}+1)(X_{d\kappa}-X_{c\kappa}+1)}. \tag{4.10}$$

Define the **associate sets** B, C of W_k as in Step 1; sets B' and C' are defined analogously for W_ℓ. Notice that $|B| = |I_k|$, $|C| = |O_k|$, $|B'| = |I_\ell|$ and $|C'| = |O_\ell|$. Using the associate sets, equation (4.10) can be expressed as

$$\sum_{\substack{X_i\in I_k \\ R_j\in O_k}}\sum_{\substack{X_c\in I_\ell \\ R_d\in O_\ell}} W(X_i,R_j)\cdot W(X_c,R_d) = \sum_{\substack{u\in B \\ v\in C}}\sum_{\substack{u'\in B' \\ v'\in C'}} \frac{1}{\prod_{\kappa=1}^{n}(u_\kappa+v_\kappa+1)(u'_\kappa+v'_\kappa+1)}. \tag{4.11}$$

The importance of the associate sets of W_k and W_ℓ is that they have positive coordinates and they are in some sense independent of the actual coordinates of the points in X, since difference choices of the point set X give the same associate sets. We now use the upper

bound from (4.8) to upper bound the RHS of (4.11). Since the RHS of (4.11) can be interpreted as a function over $2n$ dimensional points, we define the following sets in $\mathbb{R}^{2n}$

$$\mathscr{B} := \left\{(u,u') : u \in B, u' \in B'\right\}, \mathscr{C} := \left\{(v,v') : v \in C, v' \in C'\right\}$$

and

$$\mathscr{B}' := \left\{(u,v') : u \in B, v' \in C'\right\}, \mathscr{C}' := \left\{(v,u') : v \in C, u' \in B'\right\}.$$

The pair of sets $(\mathscr{B},\mathscr{C})$ and $(\mathscr{B}',\mathscr{C}')$ allow us to express the RHS of (4.11) in two different ways as:

$$\begin{aligned}\sum_{\substack{u\in B\\ v\in C}}\sum_{\substack{u'\in B'\\ v'\in C'}} \frac{1}{\prod_{\kappa=1}^{n}(u_\kappa+v_\kappa+1)(u'_\kappa+v'_\kappa+1)} &= \sum_{\substack{\mathscr{U}\in\mathscr{B}\\ \mathscr{V}\in\mathscr{C}}} \frac{1}{\prod_{\kappa=1}^{2n}(\mathscr{U}_\kappa+\mathscr{V}_\kappa+1)}\\ &= \sum_{\substack{\mathscr{U}'\in\mathscr{B}'\\ \mathscr{V}'\in\mathscr{C}'}} \frac{1}{\prod_{\kappa=1}^{2n}(\mathscr{U}'_\kappa+\mathscr{V}'_\kappa+1)}.\end{aligned}$$

From (4.8), the RHS of the equalities above can be upper bounded as

$$\begin{aligned}\sum_{\substack{\mathscr{U}\in\mathscr{B}\\ \mathscr{V}\in\mathscr{C}}} \frac{1}{\prod_{\kappa=1}^{2n}(\mathscr{U}_\kappa+\mathscr{V}_\kappa+1)} &\le (2\pi)^{2n}(|\mathscr{B}|+|\mathscr{C}|) \text{ and}\\ \sum_{\substack{\mathscr{U}'\in\mathscr{B}'\\ \mathscr{V}'\in\mathscr{C}'}} \frac{1}{\prod_{\kappa=1}^{2n}(\mathscr{U}'_\kappa+\mathscr{V}'_\kappa+1)} &\le (2\pi)^{2n}(|\mathscr{B}'|+|\mathscr{C}'|).\end{aligned}$$

So, from the two inequalities above and (4.11) we obtain

$$\begin{aligned}\sum_{\substack{X_i\in I_k\\ R_j\in O_k}}\sum_{\substack{X_c\in I_\ell\\ R_d\in O_\ell}} W(X_i,R_j)\cdot W(X_c,R_d) &\le (2\pi)^{2n}\min\{|\mathscr{B}|+|\mathscr{C}|,|\mathscr{B}'|+|\mathscr{C}'|\}.\\ &= (2\pi)^{2n}\min\{|B||B'|+|C||C'|,|B||C'|+|B'||C|\},\\ &= (2\pi)^{2n}\min\{|I_k||I_\ell|+|O_k||O_\ell|,|I_k||O_\ell|+|I_\ell||O_k|\}\end{aligned}$$

Therefore, for an arbitrary pair W_k, W_ℓ, we have

$$\sum_{\substack{X_i \in I_k \\ R_j \in O_k}} \sum_{\substack{X_c \in I_\ell \\ R_d \in O_\ell}} W(X_i, R_j) \cdot W(X_c, R_d) \leq (2\pi)^{2n} (|I_k||O_\ell| + |I_\ell||O_k|). \tag{4.12}$$

Note that every edge (X_i, R_j) in E maps to a unique path (X_i, W_k, R_j) in the graph G'. Hence the sum of the LHS of (4.12) over all possible pairs of W_k and W_ℓ gives us the sum of the product of weights of all possible pairs of edges (X_i, R_j) and (X_c, R_d) in E. Hence from (4.9) we obtain that

$$\begin{aligned} \sum_{\substack{W_k \\ W_\ell}} \sum_{\substack{X_i \in I_k \\ R_j \in O_k}} \sum_{\substack{X_c \in I_\ell \\ R_d \in O_\ell}} W(X_i, R_j) \cdot W(X_c, R_d) &= \sum_{\substack{(X_i, R_j) \in E \\ (X_c, R_d) \in E}} W(X_i, R_j) \cdot W(X_c, R_d) \\ &= \Omega(m^2 \log^{2n} m). \end{aligned} \tag{4.13}$$

Similarly, summing the RHS of (4.12) over all pairs of W_k and W_ℓ and using the fact that $|I_k| = |W_k|$ and $\sum_{k=1}^{r} |O_k| = \sum_{j=1}^{m} |\langle R_j \rangle_D|$, we get

$$\sum_{W_k} \sum_{W_\ell} (2\pi)^{2n} (|I_k||O_\ell| + |I_\ell||O_k|) = 2 \cdot (2\pi)^{2n} \left(\sum_{j=1}^{m} |\langle R_j \rangle_D| \right) \left(\sum_{k=1}^{r} |W_k| \right).$$

Therefore, from (4.12), 4.13 and the equality above, we conclude that

$$\left(\sum_{j=1}^{m} |\langle R_j \rangle_D| \right) \left(\sum_{k=1}^{r} |W_k| \right) = \Omega(m^2 \log^{2n} m).$$

Q.E.D.

4.2 Optimality of the Balanced Binary Search Tree

In this section, we will show the optimality of the balanced binary search tree in a relaxed framework where the data structures are allowed to take a weighted sum of their canonical

subsets. Let $X := \{1, 2, \ldots, m\}$. Again, consider the set of one sided range queries: For $j \in X$, output $R_j := \{i \in X : i \leq j\}$. Let $\mathscr{D} := \{W_1, W_2, \ldots, W_r\}$ be an arbitrary data structure that answers range queries on X. Proposition 19 in one dimension reduces to:

$$\left(\sum_{k=1}^{r} |W_k|\right) + \left(\sum_{j=1}^{m} |\langle R_j \rangle_D|\right) = \Omega(m \log m). \tag{4.14}$$

To extend the lower bound above for data structures that are allowed to take weighted sums of their canonical subsets, we will reinterpret range querying in a different setting. The problem of range querying can be interpreted in a linear algebraic setting as follows: Consider the $0/1$ matrix U whose rows are indexed by the m numbers and columns are indexed by the r sets, W_k's. The (i, j)th entry is one iff the number i is a member of W_j. Consider the range query that asks for all the numbers less than or equal to the jth number. The output is a union of at most, say ℓ sets. Then, R_j, which is an m-dimensional vector with ones from the jth position onwards can be expressed as a linear combination of at most ℓ columns of U. Let $\mathbf{v}_j$ be this linear combination, i.e.

$$R_j = U\mathbf{v}_j$$

where $\mathbf{v}_j$ is a $0/1$ vector. Since there are m distinct range queries, we have $\mathbf{v}_1, \ldots, \mathbf{v}_m$ such vectors. If V is the matrix with these vectors as its columns, then our observation regarding these m range queries can be succinctly represented by the following matrix equation

$$UV = \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 \\ 1 & 1 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ 1 & 1 & 1 & \cdots & 1 \end{bmatrix} =: R, \tag{4.15}$$

where R is the lower triangular matrix with all ones on and below the diagonal and the rows of R are indexed by the numbers $m, m-1, \ldots, 1$ and the columns by $R_m, R_{m-1}, \ldots, R_1$.

Also, $U \in \{0,1\}^{m\times r}$ and $V \in \{0,1\}^{r\times m}$. Now,

$$\sum_{i=1}^{r} |W_i| = ||U||_F^2 \quad \text{and} \quad \sum_{j=1}^{m} |\langle R_j \rangle_D| = ||V||_F^2,$$

where $||A||_F$ denotes the Frobenius norm of the matrix A. So, in terms of factorizations of R as a product of U and V, proving (4.14) is equivalent to deriving a lower bound on the optimal value of the following optimization problem:

$$\begin{aligned} &\min \left(||U||_F^2 + ||V||_F^2\right) \\ &\text{subject to } UV = R, U \in \{0,1\}^{m\times r}, V \in \{0,1\}^{r\times m}. \end{aligned} \tag{4.16}$$

In order to consider data structures that may take weighted sum of their canonical subsets instead of disjoint unions, we focus on the following relaxation of the problem in (4.16):: Given an arbitrary matrix $T \in \mathbb{R}^{m\times m}$,

$$\begin{aligned} &\min \left(||U||_F^2 + ||V||_F^2\right) \\ &\text{subject to } UV = T, U \in \mathbb{R}^{m\times r}, V \in \mathbb{R}^{r\times m}. \end{aligned} \tag{4.17}$$

It is clear that a lower bound on the optimal value of (4.17), when T is taken to be R, is also a lower bound on the optimal value of (4.16). So, we first prove the following result:

THEOREM 21. *Any optimal solution (U^*, V^*) for the optimization problem in* (4.17) *satisfies:*

$$||U^*||_F^2 = \text{Trace}((T^tT)^{1/2}) \text{ and } ||V^*||_F^2 = \text{Trace}((T^tT)^{1/2}),$$

where the matrix $(T^tT)^{1/2}$ is defined to be the principal square root of the matrix T^tT [24, p .20, Theorem 1.29].

Proof. The Lagrangian dual function associated with the problem in (4.17) is defined

as [11, p. 216]

$$\inf_{U,V} L(U,V,\Lambda) = \inf_{U,V} \left(||U||_F^2 + ||V||_F^2 + \sum_{i=1}^{m}\sum_{j=1}^{m}\sum_{k=1}^{r} (T_{ij} - U_{ik}\cdot V_{kj})\lambda_{ij} \right), \tag{4.18}$$

where $\Lambda \in \mathbb{R}^{m\times m}$. The Lagrange dual problem is now defined as

$$\max_{\Lambda} \left(\inf_{U,V} L(U,V,\Lambda) \right), \tag{4.19}$$

where $\Lambda \in \mathbb{R}^{m\times m}$. Any optimal solution (U^*,V^*) for the primal problem satisfies the following inequality:

$$||U^*||_F^2 + ||V^*||_F^2 \geq \max_{\Lambda} \left(\inf_{U,V} L(U,V,\Lambda) \right).$$

From the inequality above, we see that it suffices to lower bound the optimal value of the dual problem in order to prove the required claim. Consider the function

$$\inf_{U,V} L(U,V,\Lambda).$$

Applying the optimality conditions, we take the partial derivative of $L(U,V,\Lambda)$ with respect to variables in U to get the following matrix equation:

$$\nabla_U L(U,V,\Lambda) = 2U^t - V\Lambda^t = 0. \tag{4.20}$$

Similarly, taking derivative with respect to variables in V, we get

$$\nabla_V L(U,V,\Lambda) = 2V - U^t\Lambda = 0. \tag{4.21}$$

From (4.20) and (4.21), we have

$$V\Lambda^t = 2U^t \quad \text{and} \quad U^t\Lambda = 2V. \tag{4.22}$$

Since the dual problem is convex and aims to maximize the Lagrangian dual function in (4.18) with respect to Λ, we apply the first order condition to $L(U,V,\Lambda)$ with respect to Λ to get

$$UV = T.$$

By left multiplying the first equation in (4.22) by U, we get

$$\begin{aligned} UV\Lambda^t &= 2UU^t \\ T\Lambda^t &= 2UU^t \text{ since } UV = T. \end{aligned}$$

Using the equality above and the fact that $||U||_F^2 = \text{Trace}(UU^t)$, we get

$$||U||_F^2 = \frac{1}{2}\text{Trace}(T\Lambda^t).$$

Similarly, we can show that

$$||V||_F^2 = \frac{1}{2}\text{Trace}(\Lambda^t T).$$

Therefore, for an optimal solution Λ of the dual problem, we have

$$||U||_F^2 = \frac{1}{2}\text{Trace}(T\Lambda^t), \quad ||V||_F^2 = \frac{1}{2}\text{Trace}(\Lambda^t T).$$

Since $\text{Trace}(T\Lambda^t) = \text{Trace}(\Lambda^t T)$, it suffices to show that the trace of $\Lambda^t T$ is

$$2 \cdot \text{Trace}((T^t T)^{1/2})$$

to prove the theorem. We begin by multiplying the transpose of the second equation in (4.22) with the first equation in (4.22) to obtain

$$\Lambda^t UV\Lambda^t = 4(UV)^t.$$

Since $UV = T$, we see that any optimal solution for the dual problem must satisfy:

$$\Lambda^t T \Lambda^t = 4T^t.$$

Multiplying the equality above by T from the right, we get

$$(\Lambda^t T)^2 = 4T^t T.$$

Since $T^t T$ is a positive semidefinite matrix [1], it is diagonalizable. Assuming Q to be the $m \times m$ matrix whose columns are the eigenvectors of $T^t T$ and $\gamma_1 \geq \gamma_2 \geq \cdots \geq \gamma_m$ to be the eigenvalues of $T^t T$, we can express the equality above as

$$(\Lambda^t T)^2 = 4Q^{-1}\Gamma Q$$

where Γ is the diagonal matrix with γ_k's being the kth diagonal entry. Therefore, we have

$$(\Lambda^t T) = 2Q^{-1}\Gamma^{1/2}Q,$$

where $Q^{-1}\Gamma^{1/2}Q$ is defined to be the *principle square root of* $T^t T$ whose eigenvalues are $\sqrt{\gamma_1}, \ldots, \sqrt{\gamma_m}$ and for all k, $\sqrt{\gamma_k} \in \mathbb{R}_{\geq 0}$. So,

$$\text{Trace}(\Lambda^t T) = 2\text{Trace}((T^t T)^{1/2}) = 2\sum_{k=1}^{m} \sqrt{\gamma_k}.$$

Hence, we conclude that

$$||U^*||_F^2 = \text{Trace}((T^t T)^{1/2}) \quad ||V^*||_F^2 = \text{Trace}((T^t T)^{1/2}).$$

Q.E.D.

Now, we will prove the following fact about the matrix R in (4.15).

[1] Here we use the fact that any matrix A that can be written as $A = B^t B$, is positive semidefinite.

LEMMA 22. *Let R be the special lower triangular matrix as given in* (4.15). *Then, the inverse of the matrix R, R^{-1}, is given by the bidiagonal matrix*

$$R^{-1} = \begin{bmatrix} 1 & 0 & 0 & 0\dots & 0 \\ -1 & 1 & 0 & 0\dots & 0 \\ 0 & -1 & 1 & 0\dots & 0 \\ 0 & 0 & -1 & 1\dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \dots\dots & 1 \end{bmatrix}.$$

Proof. Consider the matrix R in (4.15):

$$R := \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 \\ 1 & 1 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ 1 & 1 & 1 & \cdots & 1 \end{bmatrix}.$$

To obtain the the identity matrix I from R, we can simply subtract column j from column $j+1$, $j \in [m-1]$. These $(m-1)$ elementary column operations is equivalent to multiplying the matrix R by the following bidiagonal matrix:

$$\begin{bmatrix} 1 & 0 & 0 & 0\dots & 0 \\ -1 & 1 & 0 & 0\dots & 0 \\ 0 & -1 & 1 & 0\dots & 0 \\ 0 & 0 & -1 & 1\dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \dots\dots & 1 \end{bmatrix}.$$

Hence the matrix above is the inverse matrix of R. **Q.E.D.**

Now, we will bound the trace of $(R^tR)^{1/2}$ in the following result:

LEMMA 23. *Let R be the matrix as in* (4.15)*. The trace of the principal square root of R^tR, i.e., $(R^tR)^{1/2}$ is*

$$\text{Trace}((R^tR)^{1/2}) = \sum_{k=1}^{m} \sqrt{\gamma_k} = \Omega(m \log m),$$

where γ_k, $k \in [m]$, are the eigenvalues of the matrix R^tR.

Proof. To compute γ_k's, consider

$$(R^tR)^{-1} = R^{-1}(R^{-1})^t.$$

The inverse of the matrix R from Lemma 22 is the bidiagonal matrix

$$R^{-1} = \begin{bmatrix} 1 & 0 & 0 & 0 & \dots & 0 \\ -1 & 1 & 0 & 0 & \dots & 0 \\ 0 & -1 & 1 & 0 & \dots & 0 \\ 0 & 0 & -1 & 1 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \dots & 1 \end{bmatrix}$$

So, the matrix $R^{-1}(R^{-1})^t$ is the following tridiagonal matrix

$$R^{-1}(R^{-1})^t = \begin{bmatrix} 1 & -1 & 0 & 0 & 0 & \dots & 0 \\ -1 & 2 & -1 & 0 & 0 & \dots & 0 \\ 0 & -1 & 2 & -1 & 0 & \dots & 0 \\ 0 & 0 & -1 & 2 & -1 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & 0 & \dots & -1 \\ 0 & 0 & 0 & 0 & 0 & \dots & 2 \end{bmatrix},$$

which is obtained as a special case of tridiagonal matrices of the form

$$\begin{bmatrix} a+d & b & 0 & 0 & 0 & \dots & 0 \\ b & a & b & 0 & 0 & \dots & 0 \\ 0 & b & a & b & 0 & \dots & 0 \\ 0 & 0 & b & a & b & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & 0 & \dots & b \\ 0 & 0 & 0 & 0 & 0 & \dots & a+c \end{bmatrix},$$

by substituting $a = 2$, $b = -1$, $d = -1$, and $c = 0$. From [18, p. 27], we know that the roots of the characteristic polynomial of the matrix above is given by

$$a + 2b\cos\theta \tag{4.23}$$

where θ varies over the m zeros of the following function

$$\frac{\sin(m+1)\theta - \frac{c+d}{b}\sin m\theta + \frac{cd}{b^2}\sin(m-1)\theta}{\sin\theta}.$$

Substituting $a = 2$, $b = d = -1$, and $c = 0$ in the formula above, we obtain the following expression

$$\frac{\sin(m+1)\theta - \sin m\theta}{\sin\theta}.$$

Simplifying the formula above using the sum-to-product identity[2], we get

$$\frac{2\sin\frac{(m+1)\theta - m\theta}{2}\cos\frac{(m+1)\theta + m\theta}{2}}{\sin\theta} = \frac{2\sin(\theta/2)\cos(m\theta + \theta/2)}{2\sin(\theta/2)\cos(\theta/2)} = \frac{\cos(m\theta + \theta/2)}{\cos(\theta/2)}.$$

[2]Namely, $\sin A - \sin B = 2\sin\frac{(A-B)}{2}\cos\frac{(A+B)}{2}$

The expression above vanishes at the values

$$\left(\frac{2k-1}{2m+1}\right)\pi,$$

where $k = 1,2,\ldots m$. Substituting in (4.23), we see that the eigenvalues of $R^{-1}(R^{-1})^t$ are

$$2\left(1-\cos\left(\frac{2k-1}{2m+1}\right)\pi\right) = 4\sin^2\left(\frac{2k-1}{4m+2}\right)\pi,$$

for $k = 1,2,\ldots m$, where we use the identity $(1-\cos x) = 2\sin^2 x/2$ above. So, the eigenvalues of R^tR are

$$\gamma_k = \frac{1}{4\sin^2\left(\frac{2k-1}{4m+2}\right)\pi},$$

for $k = 1,2,\ldots m$, and, therefore, the trace of the principal square root of R^tR is

$$\text{Trace}((R^tR)^{1/2}) = \sum_{k=1}^{m}\sqrt{\gamma_k} = \sum_{k=1}^{m}\frac{1}{2\sin\left(\frac{2k-1}{4m+2}\right)\pi}$$

Since for $k = 1,\ldots,m$, the reciprocal of the sine functions is a monotonically decreasing concave function, the summation above can be lower bounded as follows:

$$\text{Trace}((R^tR)^{1/2}) = \sum_{k=1}^{m}\frac{1}{2\sin\frac{(2k-1)\pi}{4m+2}} \geq \int_1^m \frac{dx}{2\sin\frac{(2x-1)\pi}{4m+2}} = \int_1^m \frac{\csc\frac{(2x-1)\pi}{4m+2}}{2}dx.$$

Substituting $y = \frac{(2x-1)\pi}{4m+2}$ and using the fact that $\int \csc y \cdot dy = \ln|\tan(y/2)|$, we obtain

$$\text{Trace}((R^tR)^{1/2}) \geq \frac{4m+2}{4\pi}\left(\ln\left(\tan\frac{(2m-1)\pi}{(8m+4)}\right) - \ln\left(\tan\frac{\pi}{(8m+4)}\right)\right).$$

As m tends to infinity, the term $\tan((2m-1)\pi/(8m+4))$ tends to one. Therefore, we have

$$\text{Trace}((R^tR)^{1/2}) = \Omega\left(m\ln\left(\cot\frac{\pi}{(8m+4)}\right)\right).$$

From the Taylor series of the cotangent function, we know that for $m \geq 1$, $\cot\frac{\pi}{(8m+4)} =$

$\Theta(m)$. Therefore,

$$\text{Trace}((R^t R)^{1/2}) = \Omega(m \log m).$$

Q.E.D.

We note that from Theorem 21 and Lemma 23, we have a stronger conclusion than in (4.14). From (4.14), we can only infer that one of the two parameters is $\Omega(m \log m)$ whereas for data structures such as balanced binary search trees, both the parameters are $\Theta(m \log m)$. Our proof shows that

$$||U_{\text{BST}}||_F = \Theta(||U^*||_F) \text{ and } ||V_{\text{BST}}||_F = \Theta(||V^*||_F),$$

where (U^*, V^*) is an optimal solution for the problem in (4.16) and $(U_{\text{BST}}, V_{\text{BST}})$ are the matrices U and V corresponding to the balanced binary search tree. Therefore, binary search trees are optimal with respect to both the parameters: The total size of the canonical sets and the total number of canonical sets needed for covering all the query ranges. Also, our proof implies that balanced binary search trees are near optimal in a more relaxed framework where the data structures are allowed to take weighted sums of their canonical subsets.

4.3 Conclusion

In this chapter, we have shown that there is a stronger tradeoff between the sizes of canonical sets and the outputs to query ranges than the one shown by Fredman. In Section 4.2, we have given an alternate proof of Proposition 18 in one dimension. Our proof also shows the optimality of balanced binary search trees in a more general setting. A natural continuation would be to generalize this proof to higher dimensions. One can start by bounding the integrality gap between an optimal solution in the boolean setting and an optimal solution of the relaxation. In one dimension, our proof shows that this

gap is at most a constant. We also believe that the optimization problem introduced in Section 4.2 is interesting in its own right. For instance, the lower bound for the average complexity of the partial sums problem [21] in the one dimensional setting can be obtained from the lower bound on the optimization problem in (4.16).

Bibliography

[1] Pankaj Agarwal, *Range Searching*, Tech. report, 2016.

[2] Pankaj Agarwal and Jeff Erickson, *Geometric Range Searching and its Relatives*, Tech. report, 1997.

[3] A. G. Akritas, A. Strzeboński, and P. Vigklas, *Implementations of a New Theorem for Computing Bounds for Positive Roots of Polynomials*, Computing **78** (2006), 355–367.

[4] A.G. Akritas, *Vincent's Theorem in Algebraic Manipulation*, Ph.D. thesis, Operations Research Program, North Carolina State University, Raleigh, North Carolina, 1978.

[5] Prashant Batra, *On the Quality of some Root-Bounds*, Sixth International Conference on Mathematical Aspects of Computer and Information Sciences (MACIS), Nov, 2015, Berlin, Germany.

[6] Prashant Batra, Maurice Mignotte, and Doru Ştefănescu, *Improvements of Lagrange's Bound for Polynomial Roots*, Journal of Symbolic Computation **82** (2017), 19 – 25.

[7] Prashant Batra and Vikram Sharma, *Bounds on Absolute Positiveness of Multivariate Polynomials*, J. Symb. Comput. **45** (2010), no. 6, 617–628.

[8] Michael Ben-Or, *Lower Bounds for Algebraic Computation Trees*, Proceedings of the Fifteenth Annual ACM Symposium on Theory of Computing (New York, NY, USA), STOC '83, ACM, 1983, pp. 80–86.

[9] Jon Louis Bentley, *Multidimensional divide-and-conquer*, Commun. ACM **23** (1980), no. 4, 214–229.

[10] Mark de Berg, Otfried Cheong, Marc van Kreveld, and Mark Overmars, *Computational geometry: Algorithms and applications*, 3rd ed. ed., Springer-Verlag TELOS, Santa Clara, CA, USA, 2008.

[11] Stephen Boyd and Lieven Vandenberghe, *Convex optimization*, Cambridge University Press, New York, NY, USA, 2004.

[12] Gerth Stølting Brodal and Riko Jacob, *Dynamic Planar Convex Hull*, 43rd Symposium on Foundations of Computer Science (FOCS 2002), 16-19 November 2002, Vancouver, BC, Canada, Proceedings, 2002, pp. 617–626.

[13] Richard A. Brualdi and Dragos Cvetkovic, *A Combinatorial Approach to Matrix Theory and Its Applications*, 1st ed., Chapman and Hall/CRC, 2008.

[14] Bernard Chazelle and Leonidas J. Guibas, *Fractional Cascading: I. A Data Structuring Technique*, Algorithmica **1** (1986), no. 1, 133–162.

[15] George E. Collins, *Krandick's Proof of Lagrange's Real Root Bound Claim*, J. Symb. Comput. **70** (2015), no. C, 106–111.

[16] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein, *Introduction to algorithms, third edition*, 3rd ed., The MIT Press, 2009.

[17] David P. Dobkin and Richard J. Lipton, *On the Complexity of Computations Under Varying Sets of Primitives*, Journal of Computer and System Sciences **18** (1979), no. 1, 86 – 91.

[18] J. F. Elliott, *The characteristic roots of certain real symmetric matrices*, 1953.

[19] M. Fredman and M. Saks, *The cell probe complexity of dynamic data structures*, Proceedings of the Twenty-first Annual ACM Symposium on Theory of Computing (New York, NY, USA), STOC '89, ACM, 1989, pp. 345–354.

[20] M. L. Fredman, *The Inherent Complexity of Dynamic Data Structures which Accommodate Range Queries*, 21st Annual Symposium on Foundations of Computer Science (sfcs 1980), Oct 1980, pp. 191–199.

[21] M. L. Fredman, *The complexity of maintaining an array and computing its partial sums*, J. ACM **29** (1982), no. 1, 250–260.

[22] Michael L. Fredman, *A Lower Bound on the Complexity of Orthogonal Range Queries*, J. ACM **28** (1981), no. 4, 696–705.

[23] Aaron Herman and Hoon Hong, *Quality of Positive Root Bounds*, Journal of Symbolic Computation **74** (2016), 592 – 602.

[24] N. Higham, *Functions of Matrices*, Society for Industrial and Applied Mathematics, 2008.

[25] Hoon Hong, *Bounds for Absolute Positiveness of Multivariate Polynomials*, J. Symb. Comput. **25** (1998), no. 5, 571–585.

[26] Hoon Hong and Dalibor Jakuš, *Testing Positiveness of Polynomials*, Journal of Automated Reasoning **21** (1998), no. 1, 23–38.

[27] I.G.Petrovskii and O.A.Oleinik, *On the Topology of Real Algebraic Surfaces*, Izv. Akad. Nauk SSSR Ser. Mat **13** (1949), 389–402.

[28] J. R. Johnson, *Algorithms for Polynomial Real Root Isolation*, Quantifier Elimination and Cylindrical Algebraic Decomposition (Vienna) (Bob F. Caviness and Jeremy R. Johnson, eds.), Springer Vienna, 1998, pp. 269–299.

[29] J. Kioustelidis, *Bounds for the Positive Roots of Polynomials*, Journal of Computational and Applied Mathematics **16** (1986), 241–244.

[30] Przemyslaw Koprowski, Kurt Mehlhorn, and Saurabh Ray, *Corrigendum to "Faster Algorithms for Computing Hong's Bound on Absolute Positiveness"*, Journal of Symbolic Computation **45** (2017), 677–683.

[31] Joseph-Louis Lagrange, *Traité de la résolution des équations numériques de tous les degrés, Œuvres de Lagrange*, 4th ed., vol. 8, Gauthier-Villars, Paris, 1879.

[32] A. Louis and S. S. Vempala, *Accelerated Newton Iteration for Roots of Black Box Polynomials*, 2016 IEEE 57th Annual Symposium on Foundations of Computer Science (FOCS), Oct 2016, pp. 732–740.

[33] G. S. Lueker, *A Data Structure for Orthogonal Range Queries*, 19th Annual Symposium on Foundations of Computer Science (sfcs 1978), Oct 1978, pp. 28–34.

[34] Kurt Mehlhorn, *Data structures and algorithms 3: Multi-dimensional searching and computational geometry*, Springer-Verlag New York, Inc., New York, NY, USA, 1984.

[35] Kurt Mehlhorn and Saurabh Ray, *Faster Algorithms for Computing Hong's Bound on Absolute Positiveness*, Journal of Symbolic Computation **45** (2010), no. 6, 677 – 683.

[36] M. Mignotte and D. Ştefănescu, *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.

[37] Maurice Mignotte, *Mathematics for computer algebra*, Springer-Verlag, Berlin, Heidelberg, 1992.

[38] John Milnor, *On the Betti Numbers of Real Varieties*, Proceedings of the American Mathematical Society, vol. 15, 1964, pp. 275–280.

[39] Swaroop N. Prabhakar and Vikram Sharma, *A Lower Bound for Computing Lagrange's Real Root Bound*, Computer Algebra in Scientific Computing (CASC), 2016, pp. 444–456.

[40] ———, *Improved Bounds on Absolute Positiveness of Multivariate Polynomials*, Proceedings of the 2017 ACM on International Symposium on Symbolic and Algebraic Computation, ISSAC '17, ACM, 2017, pp. 381–388.

[41] Franco P. Preparata and Michael I. Shamos, *Computational Geometry: An Introduction*, Springer-Verlag, 1985.

[42] Vikram Sharma, *Complexity of Real Root Isolation Using Continued Fractions*, Theor. Comput. Sci. **409** (2008), no. 2, 292–310.

[43] J. Michael Steele and Andrew C. Yao, *Lower Bounds for Algebraic Decision Trees*, Journal of Algorithms **3** (1982), no. 1, 1 – 8.

[44] D. Ştefănescu, *New Bounds for the Positive Roots of Polynomials*, Journal of Universal Computer Science **11** (2005), no. 12, 2132–2141.

[45] Doru Ştefănescu, *A New Polynomial Bound and its Efficiency*, Computer Algebra in Scientific Computing (CASC), 2015, pp. 457–467.

[46] A. Toni, *Lower Bounds on Zero-One Matrices*, Linear Algebra and its Applications **376** (2004), 275 – 282.

[47] A. van der Sluis, *Upper Bounds for Roots of Polynomials*, Numer. Math. **15** (1970), 250–262.

[48] Panagiotis S. Vigklas, *Upper Bounds on the Values of the Positive Roots of Polynomials*, Ph.D. thesis, School of Engineering, University of Thessaly, Volos, Greece, 2010.

[49] E. C. Westerfield, *A New Bound for the Zeros of Polynomials*, The American Mathematical Monthly **40** (1933), no. 1, 18–23.

[50] Chee K. Yap, *Fundamental Problems of Algorithmic Algebra*, Oxford University Press, 2000.