Thesis

SOME PROBLEMS IN SUBLINEAR ALGORITHMS

By

Rameshwar Pratap

Thesis Supervisors : Professor Sourav Chakraborty and Professor Samir Datta

Chennai Mathematical Institute

A Thesis in Computer Science submitted
to the Chennai Mathematical Institute in partial
fulfillment of the requirements for the degree of Doctor of Philosophy

June 2014

Chennai Mathematical Institute
Plot No-H1, SIPCOT IT Park,
Padur Post, Siruseri,
Tamilnadu-603103

CHENNAI
MATHEMATICAL
INSTITUTE

Rameshwar Pratap

Plot No.H1, SIPCOT IT Park
Padur Post, Siruseri-603 103
Tamil Nadu, India
Phone: +91 - 9500 140 416
E-mail: rameshwar@cmi.ac.in

DECLARATION

I declare that the thesis entitled *"Some Problems in Sublinear Algorithms"* submitted by me for the Degree of Doctor of Philosophy in Computer Science is the record of academic work carried out by me during the period from August 2009 to June 2014 under the guidance of Professor Sourav Chakraborty and Professor Samir Datta, and this work has not formed the basis for the award of any degree, diploma, associateship, fellowship or other titles in this University or any other University or Institution of Higher Learning.

Chennai Mathematical Institute
Date: June 5, 2014

Rameshwar Pratap

CHENNAI
MATHEMATICAL
INSTITUTE

Professor Sourav Chakraborty

Plot No.H1, SIPCOT IT Park
Padur Post, Siruseri-603 103
Tamil Nadu, India
Phone: +91 - 44 - 6748 0918
E-mail: sourav@cmi.ac.in

Certificate

I certify that the thesis entitled *"Some Problems in Sublinear Algorithms"* submitted for the degree of Doctor of Philosophy in Computer Science by Rameshwar Pratap is the record of research work carried out by him during the period from August 2009 to June 2014 under my guidance and supervision, and that this work has not formed the basis for the award of any degree, diploma, associateship, fellowship or other titles in this University or any other University or Institution of Higher Learning. I further certify that the new results presented in this thesis represent his independent work in a very substantial measure.

Chennai Mathematical Institute
Date: June 5, 2014

Professor Sourav Chakraborty
Thesis Supervisor

CHENNAI
MATHEMATICAL
INSTITUTE

Professor Samir Datta

Plot No.H1, SIPCOT IT Park
Padur Post, Siruseri-603 103
Tamil Nadu, India
Phone: +91 - 44 - 6748 0948
E-mail: sdatta@cmi.ac.in

Certificate

I certify that the thesis entitled *"Some Problems in Sublinear Algorithms"* submitted for the degree of Doctor of Philosophy in Computer Science by Rameshwar Pratap is the record of research work carried out by him during the period from August 2009 to June 2014 under my guidance and supervision, and that this work has not formed the basis for the award of any degree, diploma, associateship, fellowship or other titles in this University or any other University or Institution of Higher Learning. I further certify that the new results presented in this thesis represent his independent work in a very substantial measure.

Chennai Mathematical Institute
Date: June 5, 2014

Professor Samir Datta
Thesis Supervisor

Acknowledgement

Many people have contributed directly or indirectly to my academic and professional life so far. I would like to take this opportunity to say thank you to all of them. I may not be able to mention all of them in this acknowledgement and would like to give my sincere apology to them.

First and foremost, I would like to express my sincere gratitude to my PhD advisers, Prof. Sourav Chakraborty and Prof. Samir Datta for their continuous support, motivation, patience, enthusiasm, and valuable guidance throughout my doctoral years. Their guidance helped me to proceed through the doctoral program and complete my thesis. I especially would like to thank them for giving me an opportunity to work with such independence.

Also, I would like to thank faculty members of TCS at IMSc and CMI- Professor V. Arvind, Partha Mukhopadhyay, Sourav Chakraborty, C.R. Subramanian for their wonderful lectures that helped me throughout my study. I would also like to thank my Computer Science fellows, Nikhil Balaji, Abhishek Bhrushundi, Nitesh Jha, Kumar Madhukar, Prakash Saivasan for many insightful discussions and suggestions, and otherwise.

I would like to thank all my co-authors- Prof. Samir Datta, Prof. Sourav Chakraborty, Prof. Sasanka Roy, Prof. Shubhangi Saraf, and Akshay D. Kamat for their passionate participation and inputs.

I am fortunate to have many wonderful friends during my stay in CMI. Especially, I would like to thank Rajeshwari Nair, Santosh Pattanayak, Nikhil Balaji, Abhishek Bhrushundi, Nitesh Jha, Kumar Madhukar, Pabitra Barik, Suratno Basu, Narasimha Chary, Rohith Varma, Yajnaseni Dutta, Sudeshna Kolay, Nitin Saurabh for making this journey memorable.

I would like to thank Prof. Madhavan Mukund, Dean CMI for helping me in getting all financial support that was required during conference travels. I would also thank all administrative staff of CMI for their help and cooperation during my stay.

Finally, and most importantly, I would like to thank my family members for supporting me and my decisions in every aspect. I have no words to express my heartfelt thanks to them. Nothing would have been possible without their blessings, love and continuous support. Finally, this endeavor would not have existed without support from my parents. There can not be any substitute for the unconditional support and love of my parents, who have given me complete freedom over the years.

dedicated to my parents...

Table of contents

Chapter 1

Introduction

1.1 Introduction

Recent advances in the field of telecommunications, digital sensors, geographic information systems, banking, biology, finance, enterprise software, internet, artificial intelligence etc. have created massive collections of data. Extracting useful information from these huge data sets is beneficial for business, science, government and society. Various business domains are highly dependent on capturing and analyzing these massive data sets and cannot operate otherwise. For example: Various search engine companies like Google, Microsoft and Yahoo! store and index billions of web pages in their databases in order to provide fast text search; NASA's World Wind and Google Earth consolidate massive collection of satellite images to allow their users to explore Earth and its terrain; Telecommunication companies like AT&T and Vodafone evaluate terabytes of phone call records in order to generate bills of the consumers [51]; Various weather forecasting agencies simulate petabytes of atmospheric data in order to predict the weather forecast [63]. The above given examples are only a sample of various applications which require processing massive data sets.

The use of these massive data sets touches almost every aspects of our life everyday. For example:

- Search engine companies like Google, Yahoo! and Microsoft have created an entirely new paradigm in business by capturing trillions of bytes of data from the web and producing various kinds of derived data (in the form of services like satellite images, driving directions, maps, image retrieval) that are useful in many ways. These derived data are highly beneficial to the society and have transformed the way how people can easily find, access and implement this information in their day to day life.

- Consumer product companies and retail organizations collect data from all over the place like Facebook's comments, Twitter's trending tags, Google's PageRank. They use these large data sets to get an unprecedented view of customer behaviors, customer preferences, prices that different customers are willing to pay for the same product, inventory that need to be maintained in order to fulfill the varying customer demand, and so on and so forth. Companies like Amazon, eBay and Flipkart have used these techniques to develop various services like appealing recommendations, location based pricing and predictive demand forecast.

The amount of data in our world is increasing at a very large scale. It is believed that it increases by 70% to 100% every year. For many decades now, managing these rapidly increasing large volumes of data has been a big challenge. In the past, this challenge was managed by making processors faster using Moore's law, which talked about the resources required to process increasing volumes of data. But now, the situation has changed drastically. Data volumes are increasing at a much faster rate as compared to the computer resources and the processor's speed has remained more or less static.

The first major challenge is the speed with which these huge data sets need to be processed. The larger the data set that needs to be processed, the longer it will take to analyze. There are many real life situations where we would need the result of the data analysis on immediate basis. For example: Consider the case of credit card transactions. If a fraudulent credit card transaction is suspected, then it should be checked before the transaction gets completed, and ideally, the transaction should not get completed at all. In order to do so, a full analysis of the user's purchase history is required at the time of every transaction, but this is not feasible due to real time constraints. Thus, some sort of partial results related to the user's purchase history needs to be maintained in advance, so that any fraudulent transaction could immediately get identified.

The second major challenge is the memory constraint. In many real life situations, the data itself is too large to fit in the given memory, and one may have challenges to perform the computation on these big data sets with the given limited memory. For example: Spatial databases of geographic information systems (GIS) store terabytes of data about the whole earth. However, the challenge is to provide services so that one can browse comprehensive, high resolution maps using conventional computers which are restricted by the limited memory.

Another major challenge is the storage. In some cases, the size of the data set is so huge that it cannot be stored in a single machine. Thus, the data sets have to be stored in a distributed manner. In this model, if two or more systems want to compute a function on the distributed data sets, they need to communicate by sending messages to each other. But since communication is expensive, the goal is to minimize the number of bits transferred.

We can broadly classify the above mentioned challenges which occur during analysis of these massive data sets in following three categories:

- In this scenario, the data size is so massive that we cannot afford to read the whole data in order to analyze it in real time. Hence, there is a need to develop algorithms which read a very small fraction of the input and conclude about the whole input. We use randomization techniques to develop such algorithms which are called as *sublinear time algorithms*.

- In this scenario, the input data is too huge to fit in the given memory. Thus, it is desirable to develop algorithms which need to work with one or few passes over the data, and use space less than linear in the input size. These kinds of algorithms are called as *sublinear space algorithms*.

- In this scenario, the data is too huge to store in a single machine. Thus, we have to store the data in multiple machines and design algorithms that jointly perform the computations on the union of these distributed data sets. Here, the goal is to minimize both - the number (rounds) of communications as well as the size (number of bits transferred) of communications. Such algorithms are called as *sublinear in communications algorithms*.

In this thesis, we will be focusing on the first two types of scenarios, *i.e.*, sublinear time algorithms and sublinear space algorithms. We will discuss briefly about them in next two subsections.

1.1.1 Organization of the thesis

The thesis consists of 5 chapters. Chapter 1 (this chapter) is preliminary in nature and is intended to introduce the basic concepts used in this thesis. In subsections 1.1.2 and 1.1.3 of this chapter, we present a brief introduction, motivation, and some examples about sublinear time and sublinear space algorithms. In Section 1.2 of this chapter, we present a brief summary of our results (which are well explained in details in the chapters $2, 3,$ and 4). In chapters 2 and 3, we discuss about sublinear time algorithms for two problems which we considered in the property testing framework. In Chapter 4, we discuss about a sublinear space algorithm for a problem which we considered in the Logspace model of computation. Finally in Chapter 5, we provide a summary of the whole discussion.

1.1.2 Sublinear time algorithm

The study of *sublinear time* algorithms has its roots in the study of massive data sets. Internet traffic logs, financial transaction statements, social network data, etc. are examples of modern data sets whose volumes have increased by the unprecedented scale in recent years, and continue to grow rapidly. In general, an algorithm that takes an input of size n and performs computation with running time polynomial in n is considered to be efficient. Typically, in order to output the correct answer, the algorithm has to read the entire input, so the running time of that algorithm is at least linear in the size of that input. But in practice, while dealing with these huge data sets, even linear time algorithms are considered to be very slow. Hence, there is a desire to develop algorithms that read a very small fraction of the input and conclude about the whole input. Now, the challenge is to develop algorithms that operate in less than linear time by reading a small fraction of the input. At first glance, this seems like an impossible goal. However, if we use randomization techniques and allow for error and/or approximations, then such amazing algorithms do exist. The use of randomization techniques for designing algorithms that deal with large data sets can lead to vast improvements in running time, as compared to the running time of any deterministic algorithm for the same problem. The kind of approximation depends on the computation model that we consider in order to solve the problem.

In order to understand the various types of approximations, let us consider two simple examples of sublinear time algorithm:

- Let $w \in \{0,1\}^n$. Then, the problem is to determine the fraction of 1's in w. In order to determine the fraction exactly, one has to read every bit of w. However, if we want to determine an approximate answer with an additive error of ϵ, it is enough to read only $\theta(\frac{1}{\epsilon^2})$ bits. Also, it is enough to read only $\theta(\frac{1}{\epsilon^2\delta})$ bits in order to get an approximation with a multiplicative error of $(1 \pm \epsilon)$, where δ is a lower bound on the actual fraction of 1's in w.

- Let $w \in \{0,1\}^n$. Then, the problem is to decide whether w is the all 0 string. Again, in order to determine this exactly, one has to read all the bits of w. Consider the case in which we relax the exact decision problem by introducing a distance parameter $(0 < \epsilon \leq 1)$, the goal being to design a randomized algorithm which reads a very small fraction of the input and does the following, with high probability:

  – Algorithm *Accepts*, if w is all 0 string,

  – Algorithm *Rejects*, if w contains at least ϵn many 1's.

  It is easy to distinguish between the two cases (with high probability) by reading only $\theta(\frac{1}{\epsilon})$ bits of w.

The two above mentioned problems give different flavors of approximations in sublinear time algorithms. The first one *approximates the value* while the second one *approximates the decision*. In this thesis, we will be talking about the second type of problems which are known as *property testing* [44, 43, 70] problems. Here, we study sublinear time algorithms in the *property testing* framework. Property testing is an alternative notion of *approximation* for decision problems. In this framework, we relax the decision version of the problem by introducing a *promise* in the input.

Property testing

Property testing has been studied in several areas of Computer Science, Mathematics and Statistics. A number of property testing algorithms have been developed for various types of properties including graph properties, algebraic properties, geometric properties, string properties, statistical properties, clustering, properties of boolean functions, etc.

In general, we define a property testing problem over a family of objects. We encode objects by strings from $\{0,1\}^n$. Given an object $\mathcal{O}$ and a property $\mathcal{P}$, we define the distance of the object $\mathcal{O}$ from the property $\mathcal{P}$ as follows:

$$dist_{\mathcal{P}}(\mathcal{O}) = min_{\mathcal{O}' \in \mathcal{P}}\{dist(\mathcal{O}, \mathcal{O}')\},$$

where $dist(\mathcal{O}, \mathcal{O}')$ is the Hamming distance[*] between $\mathcal{O}$ and $\mathcal{O}'$. We say that $\mathcal{O}$ is ϵ-far (for a given distance parameter $0 < \epsilon \leq 1$) from satisfying $\mathcal{P}$, if $dist_{\mathcal{P}}(\mathcal{O}) > \epsilon$.

Now, for a given distance parameter ϵ, the goal is to design a randomized algorithm which reads only a small fraction of the input and distinguishes whether the input satisfies a certain predetermined *property*, or is "ϵ-far" from satisfying it. Formally, we have the following definition:

Definition 1.1. *In property testing, the goal is to query a very small fraction of the input and decide whether the input has a certain property or is "far" from satisfying it. Let $x \in \{0,1\}^n$ be a given input string. Then, a property testing algorithm, with query complexity $q(|x|)$ and proximity parameter ϵ (where $0 < \epsilon \leq 1$) for a decision problem L, is a randomized algorithm that makes at most $q(|x|)$ queries to x and distinguishes between the following two cases:*

- *if $x \in L$, then the algorithm Accepts x with probability at least $\frac{2}{3}$,*
- *if x is ϵ-far from L, then the algorithm Rejects x with probability at least $\frac{2}{3}$.*

Here, "x is ϵ-far from L" means that the Hamming distance between x and any string in L is at least $\epsilon|x|$. A property testing algorithm is said to have *one-sided error* if it satisfies the stronger condition that the accepting probability for instances $x \in L$ is 1 instead of $\frac{2}{3}$. We can easily boost the probability of success by repeating the experiment.

In property testing, the goal is to design a property testing algorithm whose query complexity and running time is sublinear in the input size. The best case is when the query complexity is independent of the input size. In that case we say that the property is *testable*.

[*]The Hamming distance between two strings of equal length is the number of positions at which the corresponding bits are different.

History

Property testing of functions was first formally introduced by Rubinfeld and Sudan [73] in the context of *program testing*. In program testing, the goal is to test whether a given program computes a specified function. This model is applicable to the theory of program testing [73, 28, 72], and also in practice where the programmer may first choose to test whether the given program satisfies the predetermined properties that are known to be satisfied by the function it computes.

The concept of property testing also naturally emerges out from probabilistic checkable proofs (PCPs). In this setting, property testing is used to test whether the function is a codeword of a specific code. This paradigm was introduced in [14]. It is also used in testing a wide variety of codes like Hadamard codes [12, 25, 26, 23, 76], long codes [47, 48, 24, 76], and codes defined by low degree polynomials [14, 15, 13, 40, 12].

Motivation

Let us now understand the motivation behind studying property testing algorithms.

- In certain cases, with large input size, we need to decide whether the input satisfies a certain predetermined property. Here, reading the entire input takes linear time, which is too large. Thus, in such cases, we could use the efficient property testing algorithm to approximate the decision.

- In other cases, the input size is not too large, but the property that we need to decide is NP-hard. Consider a case where a graph is given as an input and we need to decide whether the graph is 3-colorable (NP-hard problem). The running time of the algorithm that decides 3-colorability is exponential in the input size. For some NP hard problems like k-colorability and triangle freeness (see [68, 8, 74]) a constant (*i.e.*, independent of the input size) query property testing algorithm exists. In such cases, property testing algorithms might be quite useful.

- For cases where the input is not too large and the problem is not hard, it might be useful to get an approximate answer by running a property testing algorithm, as it is much more efficient than the corresponding exact algorithm.

- Property testing algorithms can be used to boost the running time of exact algorithms. In order to do so, we first run the property testing algorithm as a pre-processing step. If the input is far from satisfying the property, then the property tester gives indications (with high probability) towards this fact, and in some cases it also outputs a witness confirming that the input does not satisfy the property. Thus, we need to run the exact algorithm only when the input seems close to satisfying the property.

In this thesis, we consider two different problems in the property testing framework and give sublinear time algorithms for their solution. We briefly discuss their background and motivation, and converge on the main results in Subsection 1.2.1.

1.1.3 Sublinear space algorithm

There are many real life scenarios like data streams, network traffic, sensor networks and satellite transmission that generate terabytes of data. In most of these scenarios, the memory is not even enough to store the entire data. Moreover, these data sets are growing at a very fast rate. As compared to that, the available memory is not growing at an adequate rate so that the data can be processed easily. Hence, there is a desire to develop algorithms that can perform computation on these huge data sets using the limited available memory (usually sublinear in the input size).

Formally, in sublinear space algorithms, we assume that the Turing Machine (or algorithm) has a read-only input tape (like a DVD or a database), a sublinear working space (memory or RAM) and a write-only output tape (like a printer). A Turing machine reads the data from the input, performs the computation on the working space, and writes the output/result on the output tape.

There are many computational models in which sublinear space algorithms for a wide variety of problems have been studied. The number of passes that the algorithm makes on the input (in order to read the input) can be either bounded or unbounded. The number of passes on the input is bounded when the data arrives at online fashion with very fast speed. In those cases, the algorithm can make only a limited number of passes on the input in order to read it. The *Streaming model* is a notable example of a sublinear space model which has been studied extensively [9, 64]. This model has many applications in real life scenarios such as databases and networking, where the data arrives at a high speed and the time and memory required to process a chunk of the data are severely limited.

We will consider the case where the Turing machine is allowed to make an unbounded number of passes on the input, but is not allowed to use more than $O(\log n)$ (where the size of the input is n) working space. In the sublinear space model of computation, $O(\log n)$ working space is considered to be the "gold standard" working space. Here, we will be focusing on this gold standard space model, also known as *Logspace* model of computation.

Let us first understand what is so specific about the Logspace. We briefly discuss it as follows:

- Logspace is required so that one can at least address any position of the input. In most of the cases, $O(\log n)$ space is used to have multiple (constantly many) pointers to positions of the input. Logarithmic space is also sufficient to hold a logarithmic number of Boolean flags. Many basic Logspace algorithms effectively use the memory in this way.

- Logspace is enough to count up to n in binary.
- Logspace is the largest amount of space which guarantees that the time taken for performing computation cannot be greater than the polynomial time.

There are many natural problems which can be computed in Logspace. For example:

- Given two n bit numbers, many basic arithmetic operations like addition, subtraction, multiplication, and division can be computed in Logspace [50].
- Given a list of n numbers, sorting of the list can be done in Logspace.
- Given a linked list of n nodes, many basic list operations like inserting an element into the list, deleting an element from the list, merging two lists, checking whether the list contains a loop, can be done in Logspace.

In what follows, we discuss some fundamental complexity classes around the Logspace computation model and their natural *complete* problems.

Logspace Complexity

L is the class of problems that can be solved by a *deterministic* Turing machine using $O(\log n)$ space. Similarly, we denote by NL the class of problems that can be solved by a *non-deterministic* Turing machine using $O(\log n)$ space. DIRECTED REACHABILITY is a complete problem for NL. In DIRECTED REACHABILITY, the input is a *directed* graph and two designated vertices s and t, and the goal is to decide whether t is reachable from s or not. Let DSPACE$(\log^2 n)$ denote the class of problems which can be solved by a *deterministic* Turing machine using $O(\log^2 n)$ space. Then,

Theorem 1.2. ***(Savitch)*** NL$\subseteq$ DSPACE$(\log^2 n)$

SL is the class of decision problems that can be solved by a *symmetric non-deterministic* Turing machine that uses $O(\log n)$ space. A *non-deterministic* Turing machine is *symmetric* if whenever it can make a transition from a state s_i to a state s_j, then the transition from s_j to s_i is also possible. Many interesting problems fall in the complexity class SL. For example: UNDIRECTED REACHABILITY is a complete problem for SL [57]. In UNDIRECTED REACHABILITY, given an *undirected* graph and two designated vertices s and t, the goal is to decide whether s and t are connected by a path. A result by Omer Reingold proves that UNDIRECTED REACHABILITY is in L [67]. Thus, this result also implies that SL=L.

In this thesis, we solve a problem in the Logspace computation model. We briefly present the background, motivation, related work, and main results in Subsection 1.2.2.

1.2 Contributions of this thesis

1.2.1 In sublinear time algorithms

In sublinear time algorithms, one is allowed to read only a small fraction of the input and using some randomization techniques, the algorithm provides an approximate answer to the problem. Property testing, an alternative notion of approximation for decision problems, has been applied to give sublinear time algorithms for problems in a wide range of areas including graph theory, algebra, computational geometry, statistical distribution analysis, boolean function analysis, cluster analysis. Here, we consider two different types of problems in the property testing framework and give sublinear time algorithms for their solutions.

The first problem that we considered belongs to the areas of computational geometry and cluster analysis. The second problem belongs to the areas of distribution analysis and graph theory. In the following two subsections, we briefly discuss the problem statements, their motivation, related work and main results.

Testing geometric properties

Property testing algorithms have been studied for various problems in the area of discrete and computational geometry ([36], [35] and [6]). Here, we consider one of the most fundamental problems in this area which is known as clustering ([56], [52] and [10]).

Clustering is a common problem that arises in the analysis of large data sets. Clustering has a wide range of applications in areas like Biology, Computer Science, Business, Market analysis, and Medicine. For example: in the field of Biology it has applications in the areas like human genetic analysis, DNA and proteins analysis, medical image processing; in the field of Computer Science it has applications in the areas like World Wide Web, data mining, pattern recognition, social network analysis, search engine algorithms; in the fields of Business and Markets it has applications in market research, online retailing, etc. These examples are only a sample of numerous applications where clustering is extensively used.

In a typical clustering problem, we consider a set of n input points in d dimensional space and our goal is to partition the points into k clusters. We select a center point for each cluster and consider the distance from each point to the center of that cluster. In the k-center problem, our goal is to minimize the maximum of these distances. Computing k-center clustering is NP-hard: even for 2 clusters in a general Euclidean space (of dimension d); and also for an arbitrary number of clusters, even on the plane.

The corresponding decision version of the clustering problem would be the following: given a set of n points in $\mathbb{R}^d$, the problem is to decide whether all the points are contained in k unit balls or not. In property testing, we relax the exact decision problem by introducing a proximity parameter ϵ (where, $0 < \epsilon \leq 1$) in the input. Now, the goal is to decide whether all the points are contained in k unit balls, or any k unit balls contain at most $(1-\epsilon)$ fraction of points. The aim is to minimize the number of queries to the input. In [32], we considered the clustering problem in the property testing framework and give a sublinear time algorithm to solve it. We formally state the problem as follows:

Problem 1. *Given a set of n points in $\mathbb{R}^d$ as input with the proximity parameter ϵ (where $0 < \epsilon \leq 1$), the goal is to design a randomized algorithm (tester) that distinguishes between the following two cases (with high probability) while minimizing the number of queries to the input:*

- *all the points are contained in k unit balls,*
- *any k unit balls contain at most $(1-\epsilon)n$ points.*

We divide the above problem into two cases: the first one is the single object case, *i.e.* for $k = 1$; and the second one is for any $k > 1$. We now discuss both cases:

Case 1: For $k = 1$, we present a simple property testing algorithm which makes only a constant number (for a constant that depends on ϵ and d, and is independent of n) of queries to the input. Hence, with this algorithm we prove that the mentioned property is *testable*. While the algorithm is very simple, the proof of correctness is a little involved, and uses the Helly's theorem [49]. This theorem states that if a family of convex sets in $\mathbb{R}^d$ has a non-empty intersection for every $d+1$ sets, then the whole family has a non-empty intersection. In fact, since Helly's theorem also works for symmetric convex bodies[†], we can solve the above problem for any symmetric convex body instead of just a unit radius ball. Thus, we prove the following:

Theorem 1.3. *Let A be a symmetric convex body. If S is a set of n points in $\mathbb{R}^d$ as input with the proximity parameter ϵ (where, $0 < \epsilon \leq 1$), then there is an algorithm $\mathcal{A}$ that randomly samples $O(\frac{d}{\epsilon^{d+1}})$ many points and*

- *$\mathcal{A}$ accepts, if all the points in S are contained in a translated copy of A,*
- *$\mathcal{A}$ rejects with probability $\geq 2/3$, if any translated copy of A contains at most $(1-\epsilon)n$ points.*

The running time of $\mathcal{A}$ is $O(\frac{d}{\epsilon^{d+1}})$.

Case 2: For $k > 1$, we conjecture that a theorem similar to Helly's would also hold. The absence of a theorem analogous to Helly's theorem is the major obstacle to solve the problem exactly in the $k > 1$ case.

We formally state the conjecture as follows:

[†]A convex body S is called symmetric if it is centrally symmetric with respect to the origin, i.e. a point $v \in \mathbb{R}^d$ lies in S if and only if its reflection through the origin $-v$, also lies in S.

Conjecture 1.4. *For every α (where $0 < \alpha \leq 1$), there exists $\beta = \beta(\alpha, k, d)$ with the following property. Let $C_1, C_2, .., C_n$ be convex sets in R^d, $n \geq k(d+1)$, such that at least $\alpha.\binom{n}{k(d+1)}$ of the collection of subfamilies of size $k(d+1)$ are pierced* [‡] *at k points, then at least βn sets are pierced at k points.*

Assuming the above conjecture, we obtain an algorithm similar to the one stated in Theorem 1.3 that also works for the generalized k-object setting.

We are unconditionally able to solve a *weaker* version for the k object problem. In this version, we assume that the clusters are translated copies of a bounded geometric object, instead of being just unit radius balls. Our solution works for fixed k and d, and for $\epsilon \in (\epsilon', 1]$, where $\epsilon' = \epsilon'(k, t)$ (where t is a constant which depends on the shape of the geometric object).

In a related work, Alon *et al.* [6] presented a testing algorithm for (k, b)-*clustering*. A set of points is said to be (k, b)-*clusterable* if it can be partitioned into k *clusters*, where radius (or diameter) of every cluster is at most b. Section 5 of [6] presents a testing algorithm for radius cost under the L_2 metric. The analysis of this algorithm can be easily generalized to any metric where each cluster is determined by a *simple* convex set (a convex set in $\mathbb{R}^d$ is called *simple* if its VC-dimension is $O(d)$).

For testing 1-center clustering, our result and the result from [6] yield constant query testing algorithms. Although the two results have incomparable query complexity (in terms of number of queries depending on ϵ), for testing k-center clustering, we give a *weaker* query complexity algorithm that works for fixed k and d, and for $\epsilon \in (\epsilon', 1]$ where $\epsilon' = \epsilon'(k, t)$ (where t is a constant that depends on the shape of the geometric object). Alon *et al.* have used the sophisticated VC-dimension technique while we have used Helly-type results.

Our algorithms can also be used to find an *approximately good* clustering. This is also known as clustering with outliers/anomalies. Here, we have the freedom to ignore some points as outliers. We present a randomized algorithm that takes a constant sized sample from the input and outputs radii and centers of the clusters. The benefit of our algorithm is that we construct an *approximate* representation of such clustering in constant time, *i.e.,* independent of the input size.

The use of a Helly type argument in the property testing world is the main novelty of our result.

We discuss this in more detail in Chapter 2 of the thesis.

Testing uniformity of stationary distribution

A probability distribution provides all possible outcomes of a random variable and summarizes these outcomes by indicating the probability of each of them. In many real life scenarios, one

[‡]A family of sets is called n-pierceable if there exists a set S of n points such that each member of the family has a non-empty intersection with S.

can view data as the output of some probability distribution function. This approach might be very useful while dealing with large data sets like in the analysis of stock market data, in the traffic analysis of massive networks, and in the analysis of range queries on spatial databases. In such cases, it might be unfeasible to read or store the whole data and one may only query a very small fraction of the data. Also, in order to understand the properties of the data, one has to understand the properties of the underlying distribution. Some of these properties are "local" in nature that depend on one or very few domain elements which occur with a high probability. On the contrary, some properties are "global" in nature, depending on the distribution as a whole, and not on a few domain elements.

Testing various properties of distributions like closeness of two discrete distributions, closeness to an explicit given distribution [22, 21], whether the distribution has a high entropy [19], and whether two distributions are independent [20] have been thoroughly studied under the property testing framework. In this framework, we assume that the testing algorithm receives independently identically distributed (*iid*) samples from the underlying distribution.

Given two probability distributions, an interesting problem is to decide whether they are "close" or "far", under some distance measure norms. As mentioned above, testing closeness of two distributions [22, 21] (in property testing framework) is a well studied problem. In [30, 31], we consider a related problem, where one of the distributions is the uniform distribution and the other distribution is obtained by generating a random walks on a directed graph. Our property testing algorithm decides whether the stationary distribution obtained by a random walk is uniform, or such that at least some fraction of edges must be reoriented so that the resulting stationary distribution of the random walk is uniformly distributed.

Random walks on *undirected* graphs have been thoroughly studied and well understood [58]. They are closely related to *spectral graph theory* [34], which characterizes many properties of random walk on undirected graphs and has importance in both theoretical as well as practical scenarios. On the other hand, there aren't many similar studies for random walks on a directed graph. Here, we focus on a random walk in a directed graph.

A random walk on a directed graph generates a Markov chain on the vertices of the graph. An important question that often arises in the context of Markov chains is, whether the uniform distribution on the vertices of the graph is a stationary distribution. A stationary distribution of a Markov chain is a global property of the graph. This leads to the belief that whether a particular distribution is a stationary distribution of a Markov chain depends on the global structure of that Markov chain.

In [30, 31], we investigate a "local property" for degree-Δ oriented graph[†]. If this local

[†]A directed graph $\overrightarrow{G} = (V, \overrightarrow{E})$ is called a degree-Δ oriented graph if:

- Each edge is oriented and is not bidirectional,
- For all $v \in V$, $Indegree(v) + Outdegree(v) = \Delta$.

property does not hold, then it follows that the stationary distribution obtained by a random walk cannot be uniform, even without querying the whole graph. Finding this local property is very important in the sense that testing whether a stationary distribution (which is a global property of the graph) is uniform, can be reduced to testing this local property. We formally state the problem as follows:

Problem 2. *Let $\overrightarrow{G} = (V, \overrightarrow{E})$ be a degree-Δ oriented graph*[†]*. Then, we are interested in finding if there exist a "local property" such that we can reduce the testing uniformity of stationary distribution to testing the local property of the graph. If it exists, then find this local property.*

In [30, 31], we prove that for a degree-Δ oriented graph, deciding whether the uniform distribution on the vertices of the graph is a stationary distribution or not, depends on a very local property of the graph (which is contrary to normal belief). The following theorem, which is the main result of the above work, talks about that local property.

Theorem 1.5. *Let $\overrightarrow{G} = (V, \overrightarrow{E})$ be a degree-Δ oriented graph*[†]*. Then, the uniform distribution on the vertices of $\overrightarrow{G}$ is a stationary distribution (for the Markov chain generated by a random walk on $\overrightarrow{G}$) if and only if the graph has the following properties:*

1. *for all $v \in V$, $Indegree(v) \neq 0$ and $Outdegree(v) \neq 0$,*
2. *for every edge $(u, v) \in \overrightarrow{E}$, $Outdegree(u) = Indegree(v)$.*

This result also has an application to the problem of testing whether a given distribution is uniform or "far" from being uniform. More precisely, if the distribution is the stationary distribution of the *lazy random walk*[§] on a directed graph and the graph is given as an input, then how many bits of the input graph does one need to query in order to decide whether the distribution is uniform or "far" from it? We consider this problem in the **orientation model** (see [45]).

In the orientation model, the underlying graph $G = (V, E)$ is known in advance. Each edge in E is oriented (*i.e.*, directed in exactly one direction) and the orientation of edges has to be queried. The graph is said to be "ϵ-far" (where, $\epsilon \in (0, 1)$) from satisfying the property if one has to reorient at least ϵ fraction of the edges to make the graph have the property. Here, the goal is to design a property tester that distinguishes between the two cases: when G satisfies the property, and when G is ϵ-far from satisfying it. And the task should be performed while minimizing the number of queries to the graph.

We reduce the problem of testing (in the orientation model) whether the stationary distribution obtained by lazy random walk on degree-Δ directed graph is uniform to testing whether the

[§]A lazy random walk (starting from a particular vertex) on a directed graph is a random walk in which at each time step, the walk stays where it is with probability $\frac{1}{2}$ or moves according to the usual random walk. Moreover, it converges to a unique stationary distribution.

graph is Eulerian. Testing Eulerianity is a well studied problem in orientation model. In [42], Fischer *et al.* studied the problem of testing whether an oriented graph $\overrightarrow{G}$ is Eulerian. And using their result, we obtain bounds on the query complexity for testing whether the stationary distribution is uniform.

The result holds only for graphs where the in-degree plus the out-degree of all the vertices are same. It would be interesting to see if one can make a similar statement for more general graphs.

We discuss this in more detail in Chapter 3 of the thesis.

1.2.2 In sublinear space algorithms

Computing bits of algebraic number

Algebraic numbers are roots (real or complex) of finite degree polynomials with integer coefficients. Needless to say, they are fundamental objects and play an important role in all branches of Mathematics. The equivalence between reals, with recurring binary expansions (or expansions in any positive integral radix), and rationals are easy to observe. Thus, computing the bits of fixed rationals is computationally uninteresting. However, the problem becomes interesting if we focus on irrational real numbers. For example: how hard/easy is it to compute 10^{100}-th bit of $\sqrt{2}$. Computability of such numbers heralded the birth of Computer Science in Turing's landmark paper [77], where the computability of the digits of irrationals like π, e was first addressed.

Building on the surprising Bailey-Borwein-Plouffe (BBP) formula [17] for π, Yap [80] shows that certain transcendental numbers such as π have binary expansions computable in a small complexity class like deterministic logarithmic space. Motivated by this result, we seek to answer the corresponding question of algebraic numbers. We formally state the problem as follows:

Problem 3. *Let p be a fixed univariate polynomial of degree d, having integer coefficients. Then, given n as input (in unary/binary), what is the space complexity of computing the n^{th} bit of each real root of p?*

The problem of computing the n^{th} bit is called as the "verbose" version of the problem when n is given in *unary*, whereas it is called as the "succinct" version of the problem when n is given in *binary*. Obviously, the succinct version is much harder than the corresponding verbose version. We solve the verbose version in $\mathsf{GapNC}^{1\ddagger}$, a subclass of logspace. But, for the succinct version of the problem we are unable to prove a deterministic polynomial or even a non-deterministic polynomial upper bound. We can reduce the succinct version of the problem to $\mathsf{BitSLP}^{\S}$, which is computable in a finite level in the counting hierarchy[†] [4].

As our main result [38], we are able to show that:

Theorem 1.6. *Let p be a fixed univariate polynomial of degree d, having integer coefficients. If n is given in unary, then every real roots of p can be computed in the function class* GapNC^1*, and the n^{th} bit of every real root can be computed in* TCLL[‡]*. Also, if n is given in binary, then the problem of computing the n^{th} bit of each real root of p is reducible to* BitSLP[†].

Our main observation is that approximating an irrational number can be accomplished by means other than a BBP-like series. For instance, it can be approximated by using the Newton-Raphson method. The irrationality measure for algebraic numbers can be bounded by a deep theorem of Roth [71]. In order to keep our proof simple, we replace Roth's theorem by the weaker Liouville's Theorem [46] which has an elementary proof (see e.g. [75]).

An upper bound on the verbose version of the problem can be obtained by performing iterations of Newton Raphson, each of which can be viewed as composition of polynomials, which in turn can be expressed in the form of arithmetic circuits. Since Newton Raphson converges quickly, we show that the circuit would be of polynomial size and low depth. Thus, it implies that computing bits of algebraic number problem boils down to computing bits of a polynomial size arithmetic circuit.

For the upper bound on the succinct version of the problem, we observe that iterations of Newton-Raphson method can be viewed as the ratio of two *Straight Line Programs* or SLP's[§]. Thus, the problem of computing bits of algebraic numbers boils down to the problem of computing bits of the ratio of two polynomial size SLP's. This problem has been studied in literature under the name of BitSLP [5, 4]. In [4], it is shown that how to place BitSLP in a finite level of the counting hierarchy, and we use their upper bound for our result.

Related work

Jerábek [53] also considered the problem of computing the n^{th} bit of an algebraic number. He showed that for any constant d, all real and complex roots of degree d univariate rational polynomial (given by a list of coefficients in binary) can be computed to a given accuracy by a uniform TC^0[‡] circuit. The main idea behind his approach is to compute the inverse function of the polynomial by a power series. There are two main differences between our result and the result from [53]: we solve for real roots only while he solves for complex roots also; he considers only the verbose version of the problem (whose upper bound is better than ours as, $\mathsf{TC}^0 \subseteq \mathsf{GapNC}^1$) while we solve the problem for the succinct version also.

A variant of the problem has been studied by Ben-Or, Feig, Kozen and Tiwari [27]. Given a polynomial of degree d with m bit integer coefficients and an integer μ (where μ is specified

[†] It was wrongly quoted as $\mathsf{PH}^{\mathsf{PP}^{\mathsf{PP}}}$ in our paper [38]. Allender *et al.* [4] showed that BitSLP$\in \mathsf{PH}^{\mathsf{PP}^{\mathsf{PP}^{\mathsf{PP}}}}$, we can use their upper bound for our case.

[§] SLP stands for straight-line program; which is a model equivalent to arithmetic circuits and BitSLP is problem of computing a particular bit of SLP.

in unary), they consider the problem of determining all its roots, under the promise that *all roots are real*. Under this assumption, they are able to show that approximating the roots to an additive error of $2^{-\mu}$ is in NC[‡]. Notice that while their result concerns non-constant algebraic numbers, it does not involve finding the bits of the algebraic numbers but only approximating them. Also, since they only achieve unary tolerance, their claimed upper bound of NC is not better than the $\mathsf{GapNC}^1 \subseteq$ NC. Moreover, their method does not work if the all-real-roots promise is not satisfied.

Reif and Neff [65] considered the problem of approximating (complex) roots of complex polynomials. Given a univariate polynomial of degree d with complex coefficients (with norms less than 2^m in magnitude), they considered a problem to find all the roots of the polynomial up to specified precision of $2^{-\mu}$. Although their result concerns non-constant complex algebraic numbers, they do not compute the bits of algebraic number but only approximate them. If μ is specified in unary, then in the arithmetic model for computation both their algorithms (sequential and parallel) of approximating roots of constant polynomial takes logarithmic running time (logarithmic in length of μ). The model of computation (arithmetic model of computation) used in their work is different from ours hence incomparable.

We discuss this in more detail in Chapter 4 of the thesis.

[‡]For standard complexity classes like NC, we refer the reader to any standard text in complexity such as [11] and for circuit complexity classes like TC^0, FTC^0 and GapNC^1 , we refer the reader to [78] for details. One non-standard class we use is TCLL. This is inspired by the class FOLL introduced in [18] which is essentially the class of languages accepted by a uniform version of FAC^0 circuit iterated $O(\log\log n)$ many times with an AC^0-circuit on top. We obtain a TCLL-circuit by adding a TC^0-circuit on top of the iterated block of FAC^0-circuits. The class of languages accepted by such circuits constitutes TCLL.

Chapter 2

Testing Geometric Properties

2.1 Introduction

Given a set of n points in $\mathbb{R}^d$, deciding whether all the points are contained in a unit radius ball is a well known problem in Computational Geometry. Of course, the goal is to solve this problem as quickly as possible. In order to solve this problem exactly, one has to look at all the n points in the worst case scenario. But if n is too large, an algorithm with linear running time may not be fast enough. Thus, one may be interested in "solving" the above problem by taking a very small size sample and outputting the "right answer" with high probability. In this thesis, we consider the *promise* version of this problem. More precisely, for the given *proximity parameter* ϵ (where $0 < \epsilon \leq 1$), our goal is to distinguish between the following two cases:

- all the points are contained in a unit radius ball,
- no unit radius ball contains more than a $(1 - \epsilon)$ fraction of the points.

The above *promise* problem falls in the realm of property testing (see [44], [43] and [70]). In property testing, the goal is to look at a very small fraction of the input and decide whether the input satisfies the property or is "far" from satisfying it. Property testing algorithms for computational geometric problems have been studied earlier in [36], [35] and [6]. In this thesis, we study the above problem in the property testing setting and give a simple algorithm to solve it. The algorithm queries only a constant number of points (where the constant depends on the dimension d and ϵ, but is independent of n) and correctly distinguishes between the two cases mentioned above with probability at least $2/3$. While the algorithm is very simple, the proof of correctness is a little involved, and uses the Helly's theorem [49]. This theorem states that if a family of convex sets in $\mathbb{R}^d$ has a non-empty intersection for every $d + 1$ sets, then the whole family has a non-empty intersection. In fact, since Helly's theorem also works for symmetric

This chapter reports about our work done in [32].

convex bodies, we can solve the above problem for any symmetric convex body instead of just a unit radius ball. Thus, we have the following theorem which is a restatement of Theorem 1.3:

Theorem 2.1. *Let A be a symmetric convex body. If S is a set of n points in $\mathbb{R}^d$ as input with the proximity parameter ϵ (where $0 < \epsilon \leq 1$), then there is an algorithm $\mathcal{A}$ that randomly samples $O(\frac{d}{\epsilon^{d+1}})$ many points and*

- *$\mathcal{A}$ accepts, if all the points in S are contained in a translated copy of A,*
- *$\mathcal{A}$ rejects with probability $\geq 2/3$, if any translated copy of A contains at most $(1-\epsilon)n$ points.*

The running time of $\mathcal{A}$ is $O(\frac{d}{\epsilon^{d+1}})$.

One would like to generalize the above problem for more than one object, *i.e.*, given k translated copies of the object B, the goal is to distinguish between the following two cases with high probability:

- all the n points are contained in k translated copies of B,
- any k translated copies of B contains at most $(1-\epsilon)$ fraction of points.

We would like to conjecture that a similar algorithm, as stated in Theorem 2.1, would also work for the generalized k object problem. Unfortunately, the Helly's theorem does not hold for the k object setting, but we would like to conjecture that a version of the Helly-type theorem does hold for this setting. Assuming the above conjecture, we can obtain a similar algorithm for the k object setting. We can also unconditionally solve a *weaker* version of the k object problem.

2.1.1 Connection to Clustering

We can also view this problem in the context of clustering. Clustering [56, 52, 10] is a common problem that arises in the analysis of large data sets. In a typical clustering problem, we have a set of n input points in d dimensional space and our goal is to partition the points into k clusters. There are two ways to define the cluster size:

- the maximum pairwise distance between an arbitrary pair of points in the cluster,
- twice the maximum distance between a point and a chosen centroid.

The first one is called as k-center clustering for diameter cost, and the second one is called as k-center clustering for radius cost. In the k-center problem, our goal is to minimize the maximum of these distances. Computing k-center clustering is NP-hard: even for 2 clusters in general Euclidean space (of dimension d); and also for an arbitrary number of clusters, even on a plane.

In this thesis, we assume that the cluster can be of symmetric convex shape. Given a set S of n points and a symmetric convex body A in $\mathbb{R}^d$, we say that the set of points is (k, A)-clusterable if all the points are contained in k translated copies of A. In the *promise* version of the problem, for a given proximity parameter ϵ (where $0 < \epsilon \leq 1$), our goal is to distinguish between the cases when S is (k, A)-clusterable and when it is ϵ-far from being (k, A)-clusterable. We say that S is ϵ-far from being (k, A)-clusterable if at least ϵn points need to be removed from S in order to make it (k, A)-clusterable.

We solve the above problem for $k = 1$ with a constant number of queries. For $k > 1$, we solve a *weaker* version of the problem. In order to solve the *promise* version of the problem, we have designed a randomized algorithm which is generally called as *tester*.

Our algorithms can also be used to find an *approximately good* clustering. In clustering with outliers/anomalies, we have the freedom to ignore some points outlying/anomalous, in this context we present a randomized algorithm that takes a constant sized sample from the input and outputs radii and centers of the clusters. The benefit of our algorithm is that we construct an *approximate* representation of such clustering in time which is independent of the input size.

The most interesting part of our result is that we initiate an application of a Helly-type theorem, in property testing, in order to solve the clustering problem.

2.1.2 Other related work

Alon *et al.* [6] presented a testing algorithm for (k, b)-*clustering*. A set of points is said to be (k, b)-*clusterable* if it can be partitioned into k *clusters*, where the radius (or diameter) of every cluster is at most b. Section 5 of [6] presents a testing algorithm for radius cost under the L_2 metric. The analysis of this algorithm can be easily generalized to any metric under which each cluster is determined by a *simple* convex set (a convex set in $\mathbb{R}^d$ is called *simple* if its VC-dimension is $O(d)$).

For testing 1-center clustering, our result and the result from [6] give constant query testing algorithms. Although the two results have incomparable query complexity (in terms of number of queries depending on ϵ), for testing k-center clustering, we give a *weaker* query complexity algorithm which works for fixed k and d, and for $\epsilon \in (\epsilon', 1]$ where $\epsilon' = \epsilon'(k, t)$ (where t is a constant which depends on the shape of the geometric object). Alon *et al.* used the sophisticated VC-dimension technique while we have used Helly-type results.

2.1.3 Organization of the chapter

In Section 2.2, we introduce notations, definitions and state Helly and *Helly-type* theorems that are used in this chapter. In Section 2.3, we design the tester for $(1, A)$-cluster testing for a given symmetric convex body A. In Section 2.4, we design the tester for (k, G)-cluster testing for a given geometric object G. In Section 2.5, as an application of results from sections 2.3 and 2.4, we present an algorithm to find *approximate* clusters with outliers.

2.2 Preliminaries

2.2.1 Definitions

n-piercing: *A family of sets is called n-pierceable if there exists a set S of n points such that each member of the family has a non-empty intersection with S.*

Homotheticity: *Let A and B be two geometric bodies in $\mathbb{R}^d$. A is homothetic to B if there exist $v \in \mathbb{R}^d$ and $\lambda > 0$ such that $A = v + \lambda B$ (where λ is called scaling factor of B). In particular, when $\lambda = 1$, A is said to be a* ***translated copy*** *of B.*

Symmetric convex body: *A convex body A is called symmetric if it is centrally symmetric with respect to the origin, i.e., a point $v \in \mathbb{R}^d$ lies in A if and only if its reflection through the origin $-v$ also lies in A. In other words, for every pair of points $v_1, v_2 \in \mathbb{R}^d$, if $v_1 \in v_2 + A$, then $v_2 \in v_1 + A$ and vice versa. Circles, ellipses, n-gons (for even n) with parallel opposite sides are examples of symmetric convex bodies.*

2.2.2 Helly's and Fractional Helly's Theorem

In 1913, Eduard Helly proved the following theorem:

Theorem 2.2. *(Helly's Theorem [49]) Given a finite family of convex sets $C_1, C_2, ..., C_n$ in $\mathbb{R}^d$ (where $n \geq d + 1$) such that if the intersection of every $d + 1$ of these sets is non-empty, then the whole collection has a non-empty intersection.*

Katchalski and Liu proved the following result which can be viewed as a fractional version of the Helly's Theorem.

Theorem 2.3. *(Fractional Helly's Theorem [62]) For every α (where $0 < \alpha \leq 1$), there exists $\beta = \beta(d, \alpha)$ with the following property. Let $C_1, C_2, ..., C_n$ be convex sets in $\mathbb{R}^d$ (where $n \geq d + 1$). If at least $\alpha\binom{n}{d+1}$ of the collection of subfamilies of size $d + 1$ has a non-empty intersection, then there exists a point contained in at least βn sets.*

Independently, Kalai [39] and Eckhoff [54] proved that $\beta(d, \alpha) = 1 - (1 - \alpha)^{\frac{1}{(d+1)}}$. A short proof for this upper bound can be found in [7].

2.2.3 *Helly-type* theorem for more than one piercing in convex bodies

Helly's theorem on intersections of convex sets focuses on 1-pierceable families. Danzer *et al.* [37] investigated the following Helly-type problem : If d and m are positive integers, what is the least $h = h(d, m)$ such that a family of boxes (with parallel edges) in $\mathbb{R}^d$ is m-pierceable if each of its h-membered subfamilies is m-pierceable? The following is the main result of their paper:

Theorem 2.4. *1.* $h(d, 1) = 2$ *for all* d (*where* $d \geq 1$)*;*

2. $h(1, m) = m + 1$ *for all* m*;*

3. $h(d, 2) = \begin{cases} 3d \textit{ for odd } d; \\ 3d - 1 \textit{ for even } d; \end{cases}$

4. $h(2, 3) = 16$*;*

5. $h(d, m) = \infty$ *for* $d \geq 2, n \geq 3$ *and* $(d, m) \neq (2, 3)$.

Katchalski *et al.* proved a result for families of homothetic triangles in a plane [55]. This result is similar to the intersection property of axis parallel boxes in $\mathbb{R}^d$, studied by Danzer *et al.* This result can also be considered as a Helly-type theorem for more than one piercing of convex bodies. Theorem 2.5, below, presents the main result of their paper.

Theorem 2.5. *Let $\mathcal{T}$ be a family of homothetic triangles in a plane. If any nine of them can be pierced by two points, then all the members of $\mathcal{T}$ can be pierced by two points.*

This result is best possible in the following sense:

- the bound of *nine* is tight
- similar statements do not hold for homothetic (or even translated) copies of a symmetric convex hexagon

2.3 Robust Helly for one piercing of symmetric convex body

Helly's theorem is a fundamental result in discrete geometry, describing the ways in which convex sets intersect with each other. In our case, we will focus on those subsets of convex sets whose intersection properties behave *symmetrically* in certain ways. Observation 2.6 explains this in detail. In order to design the tester for the $(1, A)$-cluster testing problem, we will crucially use this observation, the Helly's theorem and the fractional Helly's theorem.

Observation 2.6. *Let A be a symmetric convex body in $\mathbb{R}^d$ containing n points, then n translated copies of A centered at these n points have a common intersection. Moreover, a translated copy of A centered at a point in the common intersection contains all these n points.*

Lemma 2.7. *Given a set S of n points in $\mathbb{R}^d$, if every $d + 1$ (where $d + 1 \leq n$) of them is contained in (a translated copy of) a symmetric convex body A, then all the n points are contained in (a translated copy of) A.*

Proof. Consider a set $\mathcal{B}$ of translated copies of A centered at points in S. Since every $d + 1$ of the given points is contained in (a translated copy of) A, by Observation 2.6, every $d + 1$ elements in $\mathcal{B}$ has a non-empty intersection. By the Helly's theorem, all elements in $\mathcal{B}$ have a non-empty intersection. Let q be a point from this intersection. Then q belongs to every element in $\mathcal{B}$ and hence, by Observation 2.6, all the centers of the elements in $\mathcal{B}$, *i.e.*, all the n points in S, are contained in (a translated copy of) A centered at q. □

Lemma 2.8. *Let S be a set of n points in $\mathbb{R}^d$ (where $n \geq d+1$). If at least ϵn (where $0 < \epsilon \leq 1$) points are not contained in any translated copy of a symmetric convex body A, then at least ϵ^{d+1} fraction of all the $d + 1$ size subsets of S (number of such subsets is $\binom{n}{d+1}$) are not contained in any translated copy of A.*

Proof. Consider a set $\mathcal{B}$ of translated copies of A centered at points in S. Now, by the fractional Helly's theorem, for every α (where $0 < \alpha \leq 1$), there exists $\beta = \beta(d, \alpha)$ such that if at least an α fraction of $\binom{n}{d+1}$ subsets (of size $d+1$) in $\mathcal{B}$ has a non-empty intersection, then there exists a point (say p) which is contained in at least β fraction of elements of $\mathcal{B}$.

Consider a translated copy of A centered at p. By Observation 2.6, for every α (where $0 < \alpha \leq 1$), there exists $\beta = \beta(d, \alpha)$ such that if at least an α fraction of $\binom{n}{d+1}$ subsets (of size $d + 1$) in S are contained in A, then at least βn points are contained in A.

Thus, if at least $(1 - \beta)n$ points are not contained in A, then at least $1 - \alpha$ fraction of $\binom{n}{d+1}$ subsets (of size $d + 1$) in S are not contained in A. (Contrapositive of the above statement.)

Since $\beta = 1 - (1 - \alpha)^{\frac{1}{(d+1)}}$ [39, 54], choosing $1 - \beta$ as ϵ makes $1 - \alpha$ equal to ϵ^{d+1}, which are the required values of the parameters. □

Theorem 2.9. *Consider a set of n points in $\mathbb{R}^d$ ($n \geq d+1$) located such that at least ϵn (where $0 < \epsilon \leq 1$) points are not contained in any translated copy of a symmetric convex body A. If we randomly sample $\frac{1}{\epsilon^{d+1}} \ln \frac{1}{\delta}$ (where $0 < \delta \leq 1$) many sets of $d + 1$ points, then there exists a set in the sample which is not contained in any translated copy of A, with probability at least $1 - \delta$.*

Proof. By Lemma 2.8, if at least ϵn points are not contained in (any translated copy of) A, then at least ϵ^{d+1} fraction of $\binom{n}{d+1}$ sets (of size $d + 1$) are not contained in (any translated copy of) A. A set of $d + 1$ points is not contained in A with probability ϵ^{d+1}. Hence, the probability that it is contained in A is $1 - \epsilon^{d+1}$. Thus, the probability that all the sampled sets are contained in A is $\leq (1 - \epsilon^{d+1})^{\frac{1}{\epsilon^{d+1}} \ln \frac{1}{\delta}} \leq e^{-\ln \frac{1}{\delta}} = \delta$. □

Algorithm 1 is a randomized algorithm, tester, for the $(1, A)$-cluster testing problem.

```
Data: A set $S$ of $n$ points in $\mathbb{R}^d$ (input is given as black-box), $0 < \delta, \epsilon \leq 1$.
Result: Returns a set of $d + 1$ points, if it exists, which are not contained in $A$ or accepts
        (i.e., all the points are contained in $A$).
1 repeat
2     select a set (say $W$) of $d + 1$ points uniformly at random from $S$
3     if $W$ is not contained in $A$ then
4         return $W$ as witness
5     end
6 until $\frac{1}{\epsilon^{d+1}} \ln \frac{1}{\delta}$ many times;
7 if no witness found then
8     return /* all the points are contained in $A$ */
9 end
```

Algorithm 1: The $(1, A)$-cluster testing in a symmetric convex body A

This algorithm has a one sided error, *i.e.*, if all the points are contained in a symmetric convex body A then it accepts the input, else it outputs a witness with probability at least $1 - \delta$. Correctness of the algorithm follows from Theorem 2.9. Thus, in the problem of testing the $(1, A)$-clustering problem for a symmetric convex body A, the sample size is independent of the input size and hence the property is *testable*. Moreover, the tester works for all the possible values of ϵ (for $0 < \epsilon \leq 1$).

2.4 Robust Helly for more than one piercing of convex bodies

2.4.1 Helly-type results for more than one piercing of convex bodies

The following lemma says that a *"Helly-type"* result is not true for circles even for 2-piercing. The result can be easily generalized for higher dimensions also. (The proof of the following lemma was suggested by Prof. Jeff Kahn in a private communication.)

Lemma 2.10. *Consider a set of n circles in a plane. For any constant w (where $w < n$), the condition that every w circles are pierced at two points is not sufficient to ensure that all the circles in the set are pierced at two points.*

Proof. We will prove the lemma by construction. Consider a unit circle and look at any chord in it bounding an arc of angle $(180 - \phi)$ degrees (for some very tiny constant ϕ). If we extend out the chord in both directions, it is a straight line which can be viewed as the limit of a large circle (that doesn't contain the center of the original unit circle). Consider all such large limiting circles obtained by all chords which bound an arc of angle $(180 - \phi)$ degrees. Then, it is easy to see that there are no two points which pierce all the circles. However, for any constant w, any w circles are pierced by two points. To see this, take any w circles and pick two random antipodal points [*] (say r_1 and r_2) from the original unit circle.

The probability that any particular circle C is neither pierced at r_1 nor at r_2 is $\frac{2\phi}{2\pi} = \frac{\phi}{\pi}$. Thus, the probability that at least one of the w circles is neither pierced at r_1 nor at r_2 is $\frac{w\phi}{\pi}$ (by the probability union bound). Hence, the probability that all the w circles are either pierced at r_1 or at r_2 is $1 - \frac{w\phi}{\pi}$.

Thus, for a given w, by taking a sufficiently small value of ϕ (where $\phi < \frac{\pi}{w}$), we can make above probability high enough. Hence, by probabilistic arguments, we prove that there exist some two points that pierce any constant number of circles. □

Using arguments similar to the proof of above lemma, it is easy to prove that a *"Helly-type"* result for more than one piercing is also not true for a set of translated ellipsoids. Katchalski *et al.* [55] and Danzer *et al.* [37] proved a *"Helly-type"* result for more than one piercing of triangles and boxes, respectively. According to [55], a *"Helly-type"* result for more than one piercing is not true for centrally symmetric hexagons (with parallel opposite edges). A similar type of result is true for triangles and pentagons (with pair of parallel edges) which are not symmetric convex bodies. Thus, among symmetric convex bodies (spheres, ellipsoids and n-gons (for $n \leq 6$)), a *"Helly-type"* result for more than one piercing is possible only for parallelograms. We have following observation regarding the same:

Observation 2.11. *Let S be a set of n points in $\mathbb{R}^d$. If every set of h points (for finite possible values of h, see Theorem 2.4) in S is contained in m (where $m > 0$) translated parallelograms, then all the n points are contained in m translated parallelograms.*

Proof. Consider the set $\mathcal{B}$ of translated parallelograms centered at points in S. Since every set of h points is contained in m translated parallelograms, every h-membered subset of $\mathcal{B}$ is m-pierceable (by Observation 2.6). Thus, by Theorem 2.4, $\mathcal{B}$ is m-pierceable. Let $q_1, q_2, ..., q_m$ be m points in these m-intersections. Each element of $\mathcal{B}$ contains at least one of the q_i's (where $1 \leq i \leq m$) and, therefore, m translated parallelograms with centers as $q_1, q_2, ..., q_m$ contain all the points in S (by Observation 2.6). □

2.4.2 Fractional Helly for more than one piercing of convex bodies

We now design a *weaker* version of the tester for the (k, G)-clustering (where G is a bounded geometric object and $k > 1$). The tester works for some particular value of $\epsilon \in (\epsilon'(k,t), 1]$, where t is some constant that depends on the shape of geometric object.

[*]The antipode of a point on the perimeter of a circle is the point which is diametrically opposite to it.

We state the following conjecture for more than one piercing of convex bodies.

Conjecture 2.12. *For every α (where $0 < \alpha \leq 1$), there exists $\beta = \beta(\alpha, k, d)$ with the following property. Let $C_1, C_2, .., C_n$ be convex sets in R^d, $n \geq k(d+1)$, such that at least $\alpha.\binom{n}{k(d+1)}$ of the collection of subfamilies of size $k(d+1)$ are pierced at k points, then at least βn sets are pierced at k points.*

Lemma 2.13. *If Conjecture 2.12 is true, then we have the following: Consider a set of n points in $\mathbb{R}^d$ (where $n \geq k(d+1)$). If at least ϵn (where $0 < \epsilon < 1$) points are not contained in any k translated copies of symmetric convex body A, then at least $(1-\alpha)$ fraction of $\binom{n}{k(d+1)}$ subsets of size $k(d+1)$ are not contained in any k translated copies of A.*

Proof. Proof of this lemma is similar to the proof of Lemma 2.8. □

Now, we prove a *weaker* version of Conjecture 2.12. We show that for bounded geometric objects, a weaker version of fractional Helly for more than one piercing is true. We use the *greedy* approach to prove the same. We prove it for some $\epsilon \in (\epsilon', 1]$, where $\epsilon' = \epsilon'(k, t)$ (where t is a constant that depends on the shape of the geometric object). The result is true only for constant k and d.

Lemma 2.14. *Consider k translated copies of a geometric object G and a set of n points in $\mathbb{R}^d$ (for constant k and d). Then there exist $\epsilon' = \epsilon'(k, t)$ (where $\epsilon'(k, t) = 1 - \frac{1}{2(t+1)(k+1)}$, t is a constant that depends on the shape of the geometric object) such that for all $\epsilon \in (\epsilon', 1]$, if at least ϵn points are not contained in any k translated copies of G, then there exist at least $\Omega(n^{k+1})$ many witnesses of $k+1$ points which are not contained in any k translated copies of G.*

Proof. We say a geometric object G is *best* if it encloses the maximum number of points from the given set of n points. Now, we start with such a best object. Let us say the best object contains at least $c_0(1-\epsilon)n$ points (where $0 < c_0 \leq 1$). Now draw an object, L_G, concentric and homothetic with respect to G, having a scaling factor of $2 + \varepsilon$ (for $0 < \varepsilon \ll 1$, see the definition of Homotheticity in Subsection 2.2.1 where $v = 0$ and $\lambda = 2 + \varepsilon$). We define $t(= \kappa^d - 1$, see Lemma 2.16) to be the number of translated copies of G required to cover the annulus between two concentric objects G and L_G. Since we started with the best object, the annulus contains at most $tc_0(1-\epsilon)n$ points. Hence, the number of points which are outside L_G is at least $\epsilon n - tc_0(1-\epsilon)n = \epsilon_1 n$, where $\epsilon_1 = \epsilon - tc_0(1-\epsilon)$. We throw away all the points in the annulus. Now, we are left with best object that contains at least $c_0(1-\epsilon)n$ points and the remaining space contains at least $\epsilon_1 n$ points.

Now, we repeat the above process on $\epsilon_1 n$ points and would keep on repeating it until every point is either deleted or contained in some translated copies of G. Thus, total number of points that we have deleted from annuli is at most $t\Sigma_{i\geq 0} c_i(1-\epsilon_i)n$ and total number of points that are inside translated copies of G is at least $\Sigma_{i\geq 0} c_i(1-\epsilon_i)n$ (where $\epsilon_0 = \epsilon$).

By construction, the total number of points inside translated copies of G and the points that have been deleted from annuli is at least n. Thus,

$$\Sigma_{i\geq 0}c_i(1-\epsilon_i)n + t\Sigma_{i\geq 0}c_i(1-\epsilon_i)n \geq n \quad (\text{where } \epsilon_0 = \epsilon).$$

$$\Sigma_{i\geq 0}c_i(1-\epsilon_i)n \geq \frac{n}{t+1}.$$

Let G_i denotes the i-th geometric object and $|G_i|$ denotes the number of points contained in it. Thus,

$$\Sigma_{i\geq 0}|G_i| \geq \frac{n}{t+1}.$$

By assumption, k translated copies of G contains at most $(1-\epsilon)n$ points.
Thus, $|G_i| \leq (1-\epsilon)n$. Since $\epsilon > 1 - \frac{1}{2(t+1)(k+1)}$,

$$|G_i| < \frac{n}{2(t+1)(k+1)}.$$

Now, our goal is to make $k+1$ buckets, $S_1, S_2, .., S_{k+1}$, from G_i's such that each bucket contains at least $\frac{n}{2(t+1)(k+1)}$ points and at most $\frac{n}{(t+1)(k+1)}$ points. We construct these buckets by adding points from G_i's until its size become at least $\frac{n}{2(t+1)(k+1)}$. Since each $|G_i| < \frac{n}{2(t+1)(k+1)}$ and $\Sigma_{i\geq 0}|G_i| \geq \frac{n}{t+1}$, this construction is possible. Thus, for a particular bucket S_i,

$$\frac{n}{2(t+1)(k+1)} \leq |S_i| \leq \frac{n}{(t+1)(k+1)}.$$

Now, choosing one point from each of the $(k+1)$ buckets gives a set of $k+1$ points as a witness, which is not contained in k translated copies of G. Thus, there are at least $\left(\frac{1}{2(t+1)(k+1)}\right)^{k+1} n^{k+1}$ $(= \Omega(n^{k+1}))$ many witnesses. □

Theorem 2.15. *Consider k translated copies of a geometric object G and a set of n points in $\mathbb{R}^d$ (for constant k and d). Then there exist $\epsilon' = \epsilon'(k,t)$ (where t is a constant that depends on the shape of the geometric object) such that for all $\epsilon \in (\epsilon', 1]$, at least ϵn points are not contained in any k translated copies of G. Now, if we randomly sample $\frac{1}{c}\ln\frac{1}{\delta}$ (where $0 < \delta \leq 1$, $c = \left(\frac{1}{2(t+1)(k+1)}\right)^{k+1}$, and cn^{k+1} is the number of witnesses, see Lemma 2.14) many sets of size $k+1$, then there exists a set in the sample which is not contained in any k translated copies of G, with probability at least $1-\delta$.*

Proof. By Lemma 2.14, if at least ϵn points are not contained in any k translated copies of G, then there exist at least cn^{k+1} many witnesses of $k+1$ points which are not contained in any k translated copies of G. A set of $k+1$ points is not contained in k translated copies of G with probability $\frac{cn^{k+1}}{n^{k+1}} = c$. Hence, the probability that it is contained in k translated copies of G is $1-c$. Thus, the probability that all the sampled sets are contained in k translated copies of G is $\leq (1-c)^{\frac{1}{c}\ln\frac{1}{\delta}} \leq e^{-\ln\frac{1}{\delta}} = \delta$. □

Similar to tester for the $(1, A)$-cluster testing problem, we present a tester (Algorithm 2) for the problem (k, G)-cluster testing. If all the points are contained in k translated copies of G then algorithm accepts the input, else it outputs a witness with probability at least $1 - \delta$. Correctness of the algorithm follows from Theorem 2.15. Thus, similar to testing $(1, A)$-clustering, this property is also *testable*. But, the tester only works for constant k and d and for $\epsilon \in (\epsilon', 1]$ (see Lemma 2.14).

```
    Data: A set $S$ of $n$ points in $\mathbb{R}^d$ (input is given as black-box), $0 < \delta \leq 1$ and $\epsilon \in (\epsilon', 1]$.
    Result: Returns a set of $k + 1$ points, if it exists, which is not contained in $k$ translated
            copies of $G$, or accepts (i.e., all the points are contained in it).
1   repeat
2       select a set (say $W$) of $k + 1$ points uniformly at random from $S$
3       if $W$ is not contained in $k$ translated copies of $G$ then
4           return $W$ as witness
5       end
6   until $\frac{1}{c} \ln \frac{1}{\delta}$ many times;
7   if no witness found then
8       return /* all the points are contained in $k$ translated copies of $G$ */
9   end
```
Algorithm 2: The (k, G)-cluster testing in geometric objects

Lemma 2.16. *Let G be a bounded geometric object. Consider another geometric object L_G, concentric and homothetic with respect to G, having a scaling factor of $2 + \varepsilon$ (for $0 < \varepsilon \ll 1$, see the definition of Homotheticity in Subsection 2.2.1). Now, the annulus obtained between two concentric objects G and L_G can be covered by $\kappa^d - 1$ translated copies of G, where κ is (ceiling of) the ratio of side length of the smallest d-cube circumscribing L_G to that of the largest d-cube (homothetic w.r.t. smallest d-cube circumscribing L_G) inscribing G.*

Proof. Let C_{L_G} be the smallest d-cube circumscribing L_G and C_G be the largest d-cube (homothetic *w.r.t.* C_{L_G}) inscribing G. Let $C_{L_G} = [0, \kappa]^d$. Now, consider the d-dimensional grid of C_{L_G} obtained by points whose coordinates are from the set $\{0, 1, 2, 3, .., \kappa\}$. One translated copy of C_G would be require to cover each of the unit d-cube from the d-dimensional grid, and hence κ^d translated copies of C_G would require to cover C_{L_G}. Thus, a covering by $\kappa^d - 1$ translated copies of C_G would be required to cover the annulus between C_{L_G} and C_G.

Now, in order to get a bound for geometric objects, in the above covering, we can replace C_{L_G} by L_G and C_G by G. Clearly, the cube covering bound would be an upper bound for geometric object covering. □

2.5 Application in Clustering with Outliers

While considering the clustering problem, we mostly assume that the data is perfectly clusterable. But a few random points (outliers, noise) could be added in the data by an adversary. For example, in the k-center clustering, if an adversary adds a point in the data which is very far from the original set of well clustered points, then in the optimum solution that point becomes center of its own cluster and the remaining points are forced to clustered with $(k-1)$ centers only. Also, it is even difficult to locate when a point becomes an outlier. For example: consider a set of points where we need to find its optimal k-center clustering. Take a point from that set and keep moving it far from the remaining set. Now, it is very difficult to locate correctly at which place that point becomes center of its own cluster and the remaining points are left with $(k-1)$-center clusters.

In this thesis, we consider clustering with outliers by ignoring some fraction of points. Thus, in the case when points are perfectly clusterable, ignoring some fraction of points does not affect the result too much, and the case when outliers are present, the algorithm has the ability to ignore them while computing the final clusters. It may seems that the ability to ignore some fraction of points makes the problem easier, but on the contrary it does not. Because the algorithm not only has to decide which point to include in the cluster, but also has to decide which point to include first. There may be two extreme approaches to solve this problem: 1) Decide which points are outliers and run the clustering algorithm; 2) Do not ignore any points, and after getting final clusters decide which ones are outliers. Unfortunately, neither of these two approaches works well. The first one scales poorly because there are exponentially many choices, and the second one may significantly change the final outcome when outliers are indeed present. This motivates the study of clustering with outliers (see[33]).

Theorem 2.9 has an application to the 1-center clustering with outliers. More precisely, for $0 < \epsilon, \delta \leq 1$, when we have the ability to ignore at most ϵn points as outliers, we present a randomized algorithm which takes a constant size sample from input and correctly output the radius and center of the *approximate* cluster with probability at least $1 - \delta$.

```
  Data: A set $S$ of $n$ points in $\mathbb{R}^d$ (input is given as black-box), $0 < \epsilon, \delta \leq 1$.
  Result: Report center and radius of cluster which contains all but at most $\epsilon n$ points.
1 Uniformly and independently, select $m = \frac{d+1}{\epsilon^{d+1}} \ln \frac{1}{\delta}$ points from $S$.
2 Compute minimum enclosing ball containing all the sample points and report its center
  and radius.
```

Algorithm 3: The 1-center clustering with outliers

Theorem 2.17. *Given a set of n points in $\mathbb{R}^d$ and $0 < \epsilon, \delta \leq 1$, Algorithm 3 correctly outputs, with probability at least $1-\delta$, a ball containing all but at most ϵn points in constant time by querying a constant size sample (constant depending on d and ϵ). Moreover, if $r_{outlier}$ is the smallest ball containing all but at most ϵn points and r_{min} is the smallest ball containing all the points, then Algorithm 3 outputs the radius r such that $r_{outlier} \leq r \leq r_{min}$.*

Proof. From Theorem 2.9, if a sample of size m is contained in a ball of radius r, then this ball contains all but at most ϵn points, with probability at least $1 - \delta$. We compute the value of r in step 2 using the Algorithm of [61], which takes $\theta(m)$ time. Thus, both the sample size and the running time of the algorithm are constant. Clearly, $r_{outlier} \leq r \leq r_{min}$. □

The problem of clustering with outliers can be generalized to k-center clustering. If Conjecture 2.12 is true, then it has an application to k-center clustering with outliers. For given $0 < \epsilon, \delta \leq 1$, ignoring at most ϵn points as outliers, we present a randomized algorithm which takes a constant size sample from the input and correctly outputs the radii and k centers of the *approximate* clusters with probability at least $1 - \delta$.

```
Data: A set $S$ of $n$ points in $\mathbb{R}^d$ (input is given as black-box), $0 < \epsilon, \delta \leq 1$.
Result: Reports $k$ centers and radii of clusters which contain all but at most $\epsilon n$ points.
1 Uniformly and independently, select $m = \frac{k(d+1)}{(1-\alpha)} \ln \frac{1}{\delta}$ points from $S$.
2 Compute $k$ minimum enclosing balls containing all the sample points and report their
  centers and radii.
```

Algorithm 4: The k-center clustering with outliers

Theorem 2.18. *Consider a set of n points in $\mathbb{R}^d$. If Conjecture 2.12 is true and $0 < \epsilon, \delta \leq 1$, then with probability at least $1 - \delta$, Algorithm 4 outputs k balls containing all but at most ϵn points in constant time by querying a constant sized sample (constant depending on k, d and ϵ). Moreover, for $1 \leq i \leq k$, if $r^{(i)}_{outlier}$ is the radius of the optimal i-th cluster by ignoring at most ϵn points as outliers and $r^{(i)}_{min}$ is the radius of the optimal i-th cluster when all points are present, then Algorithm 4 outputs the radius $r^{(i)}$ such that $r^{(i)}_{outlier} \leq r^{(i)} \leq r^{(i)}_{min}$.*

Proof. If Conjecture 2.12 is true, then from Lemma 2.13, if at least ϵ fraction of points are not contained in k translated copies of a symmetric convex body A, then at least $(1 - \alpha)$ fraction of $\binom{n}{k(d+1)}$ sets of size $k(d + 1)$ are not contained in k translated copies of A. Now, similar to 1-center clustering with outliers, a sample of size $m = \frac{k(d+1)}{(1-\alpha)} \ln \frac{1}{\delta}$ is sufficient.

The step 2 of Algorithm 4 can be computed in time $O(m^{kd+2})$ using the algorithm of Agarwal *et al.* (see Section 7.1 of [3] for details). For relatively smaller values of d, we can use the Algorithm of [2] to get a better running time ($m^{O(f(d).k^{1-\frac{1}{d}})}$, where $f(d)$ is always bounded by $O(d^{\frac{5}{2}})$). Thus, the sample size as well as the running time of the algorithm are constant. Clearly, $r^{(i)}_{outlier} \leq r^{(i)} \leq r^{(i)}_{min}$. □

2.6 Conclusion and Open Problems

In this thesis, we initiated an application of the Helly (and *Helly-type*) theorem in property testing. For the $(1, A)$-cluster testing in a symmetric convex body A, we showed that testing can be done with constant number of queries and hence proved that the property is *testable*. Alon *et al.* [6] also solved a similar problem with constant number of queries, using the combination of sophisticated arguments in geometric and probabilistic analysis. For the 1-center clustering, our result had an incomparable query complexity in relation (in terms of number of queries depending on ϵ) with the result of Alon *et al.* We stated a conjecture related to the fractional *Helly-type* theorem for more than one piercing of convex bodies. Using a greedy approach, we proved a weaker version of the conjecture which we used for testing (k, G)-clustering. We also gave a characterization of the type of symmetric convex body for which Helly-type result for more that one piercing would be true. Finally, as an application of testing result in clustering with outliers, we showed that one can find, with high probability, the *approximate* clusters by querying a constant size sample.

In this thesis, we attempted upper bounds problems for testing the $(1, A)$ clustering, and the (k, G) clustering problems. However, proving lower bounds for these problems remains open. Also, for the 1-center clustering, we studied the case when the cluster is a symmetric convex object. Proving upper bounds when the cluster is non-symmetric and/or non-convex object will be an interesting open problem.

Chapter 3

Testing Uniformity of Stationary Distribution

3.1 Introduction

Markov chains are one of the most important and well studied structures in Theoretical Computer Science. The use of Markov chains in other fields like Physics, Biology, Economics, Statistics etc. cannot be overemphasized. The most important characteristics of a Markov chain are its stationary distribution and its mixing time. It is very important to understand the structure of stationary distribution of a Markov chain. To be precise, one often wants to know if a particular distribution is a stationary distribution of a given Markov chain. Stationary distribution of a Markov chain is a global property of the graph. This leads to the belief that whether a particular distribution is a stationary distribution of a Markov chain depends on the global structure of that Markov chain.

One of the most commonly used Markov chains in Theoretical Computer Science is the Markov chain obtained by a random walk on a directed graph. In this thesis, we focus on this kind of Markov chain. We prove that contrary to normal perception, for this kind of Markov chain, if the graph is regular then whether the uniform distribution is a stationary distribution depends on a local property of the graph. The following theorem, which is the main result of this chapter (see section 3.3), is a statement about that local property. (The following theorem is a restatement of Theorem 1.5.)

Theorem 3.1. *Let $\overrightarrow{G} = (V, \overrightarrow{E})$ be a directed graph such that the total degree (i.e., Indegree(v)+ Outdegree(v)) for every vertex $v \in V$ is the same. Then the uniform distribution on the vertices of $\overrightarrow{G}$ is a stationary distribution of the Markov chain (generated by a random walk on $\overrightarrow{G}$) if and only if the graph has the following properties:*

This chapter reports about our work done in [30, 31].

1. *for all $v \in V$, Indegree(v) $\neq 0$ and Outdegree(v) $\neq 0$,*
2. *for all $(u, v) \in \overrightarrow{E}$, Outdegree(u)=Indegree(v).*

As an application of this result, we want to test whether the uniform distribution on the vertices is the stationary distribution of the Markov chain generated by a *lazy random walk* [*] on $\overrightarrow{G}$.

3.1.1 Application to Property Testing of Distributions

In property testing, the goal is to look at a very small fraction of the input and distinguish whether the input has a certain property or it is "far" from satisfying the property. If $\overrightarrow{G} = (V, \overrightarrow{E})$ is a digraph such that the total degree (*i.e.,* Indegree(v) $+$ Outdegree(v)) of every vertex $v \in V$ is the same, then we consider the Markov chain obtained by the lazy random walk on vertices of $\overrightarrow{G}$. Thus, the distribution can be given implicitly by the directed graph.

Suppose that the underlying undirected graph is known beforehand and for every edge there is a unit cost to know its orientation. Now, we want to distinguish between the following two cases:

- whether the uniform distribution is the stationary distribution of the Markov chain,
- whether more than ϵ fraction of edges have to be reoriented so that the resulting stationary distribution of the Markov chain is the uniform distribution.

The goal is to distinguish between the above cases while incurring minimum cost, that is, by knowing the orientation of minimum number of edges. This model for testing graph properties is called the **orientation model** [45]. This model has been well studied in the field of property testing. We show that the above problem can be reduced to an earlier studied problem of testing (in the orientation model) whether a directed graph is Eulerian. We use the results of [42] to obtain algorithm that incurs sublinear cost for the above problem. We present this part of our result in Section 3.4.

3.2 Preliminaries

3.2.1 Graph Notations

Throughout this chapter, we will be dealing with directed graphs (possibly with multiple edges between any two vertices) in which each edge is directed only in one direction. To avoid confu-

[*] A lazy random walk (starting from a particular vertex) on a directed graph is a random walk in which at each time step, the walk stays where it is with probability $\frac{1}{2}$ or moves according to the usual random walk. Moreover, it converges to a unique stationary distribution.

sion, we will call them **oriented graphs** because each edge is oriented and is not bidirectional. We will denote the oriented graph by $\overrightarrow{G} = (V, \overrightarrow{E})$ and the underlying undirected graph (that is, when the directions on the edges are removed) by $G = (V, E)$. For a vertex $v \in V$, the in-degree and the out-degree of v in $\overrightarrow{G}$ are denoted by $d^-(v)$ and $d^+(v)$ respectively. An oriented graph $\overrightarrow{G} = (V, \overrightarrow{E})$ is called a *degree-*Δ oriented graph if for all $v \in V$, $d^-(v) + d^+(v) = \Delta$. In this thesis, we will be focusing on *degree-*Δ oriented graphs.

3.2.2 Markov Chains

A Markov chain is a stochastic process on a set of states given by a transition matrix. Let S be the set of states with $|S| = n$. Then, the transition matrix T is an $n \times n$ matrix with entries from the positive reals; the rows and columns are indexed by the states; the (u, v)-th entry $T_{u,v}$ of the matrix denotes the probability of transition from state u to state v. Since T is stochastic, $\sum_v T_{u,v}$ must be 1. A distribution $\mu : S \to \mathbb{R}^+$ on the vertices is said to be stationary if for all vertices v,

$$\sum_v \mu(u) T_{u,v} = \mu(v).$$

If $\overrightarrow{G}$ is an oriented graph then a random walk on $\overrightarrow{G}$ defines a Markov chain, where, the states are the vertices of the graph; the probability to traverse an edge (u, v) is given by the quantity $p_{u,v} = \frac{1}{d^+(u)}$; and hence, the transition probability $T_{u,v}$ from vertex u to vertex v is $p_{u,v}$ times the number of edges between u and v.

In this thesis, we will only consider Markov chains that arise from random walks on $\overrightarrow{G}$, where $\overrightarrow{G}$ is an oriented graph. The uniform distribution on the vertices of $\overrightarrow{G}$ is a stationary distribution for this Markov chain if and only if for all $v \in V$,

$$\sum_{u:(u,v)\in\overrightarrow{E}} p_{u,v} = 1 = \sum_{w:(v,w)\in\overrightarrow{E}} p_{v,w}.$$

Note 3.2. *As mentioned above, in the rest of the chapter, we will be focusing on the Markov chain generated by a random walk on $\overrightarrow{G}$. So, any occurrence of Markov chain needs to be interpreted in a similar way.*

3.3 Structure of Graphs with Uniform Stationary Distribution

The following theorem is a rephrasing of Theorem 3.1.

Theorem 3.3. *Let $\overrightarrow{G} = (V, \overrightarrow{E})$ be a degree-Δ oriented graph, then the uniform distribution on the vertices of $\overrightarrow{G}$ is a stationary distribution of the Markov chain if and only if for all $v \in V$,* *both* $d^-(v), d^+(v) \neq 0$ *and for all* $(u,v) \in \overrightarrow{E}$*,*

$$d^+(u) = d^-(v).$$

Proof. First of all, note that the uniform distribution is a stationary distribution for $\overrightarrow{G}$, iff for all $v \in V$,

$$\sum_{u:(u,v)\in\overrightarrow{E}} p_{u,v} = 1 = \sum_{w:(v,w)\in\overrightarrow{E}} p_{v,w},$$

where $p_{u,v}$ is the transition probability from vertex u to vertex v. Since we are doing a random walk on an unweighted oriented graph, for all $(u,v) \in \overrightarrow{E}$, $p_{u,v} = \frac{1}{d^+(u)}$.

Thus, if the graph $\overrightarrow{G}$ has the property that for all $(u,v) \in \overrightarrow{E}$, $d^+(u) = d^-(v)$, then note that

$$\sum_{u:(u,v)\in\overrightarrow{E}} p_{u,v} = \sum_{u:(u,v)\in\overrightarrow{E}} \frac{1}{d^+(u)} = \sum_{u:(u,v)\in\overrightarrow{E}} \frac{1}{d^-(v)} = 1.$$

The last equality holds because the summation is over all the edges entering v (which is non-empty) and thus there are $d^-(v)$ number of items in the summation. Similarly,

$$\sum_{w:(v,w)\in\overrightarrow{E}} p_{v,w} = \sum_{w:(v,w)\in\overrightarrow{E}} \frac{1}{d^+(v)} = 1.$$

Therefore, if the graph $\overrightarrow{G}$ has the property that for all $(u,v) \in \overrightarrow{E}$, $d^+(u) = d^-(v)$, then the uniform distribution is a stationary distribution of the Markov chain.

Now, let us prove the other direction, that is, let us assume that the uniform distribution is a stationary distribution of the Markov chain. Note that, if the uniform distribution is a stationary distribution, then there is a path from u to v if and only if u and v are in the same strongly connected component of $\overrightarrow{G}$. This is because the uniform distribution is a stationary distribution if and only if for every cut $C = V_1 \cup V_2$, where $V_2 = (V \backslash V_1)$, we have

$$\sum_{(u,v)\in\overrightarrow{E},\text{ and } u\in V_1, v\in V_2} p_{u,v} = \sum_{(u,v)\in\overrightarrow{E},\text{ and } u\in V_2, v\in V_1} p_{u,v}.$$

In other words, if a stationary distribution is uniform, then it implies that every connected component in the undirected graph is strongly connected in the directed graph.

Let $v_0, v_1, v_2, \ldots, v_t$ be a sequence of vertices such that the following conditions are satisfied:

- For all $i \geq 0$, $(v_{i+1}, v_i) \in \overrightarrow{E}$,
- For all $i \geq 0$, $d^+(v_{2i+1}) = \min\left\{d^+(w) \ : \ (w, v_{2i}) \in \overrightarrow{E}\right\}$,

- For all $i > 0$, $d^+(v_{2i}) = \max\left\{d^+(w) \ : \ (w, v_{2i-1}) \in \overrightarrow{E}\right\}$.

We call such a sequence as "degree-alternating" sequence of vertices.

Claim 3.4. *Let $\{v_i\}$ be a "degree-alternating" sequence of vertices. If we define a new sequence $\{S\}$ of positive integers as following:*

- *For all $k \geq 0$, s_{2k}=$d^-(v_{2k})$,*
- *For all $k \geq 0$, s_{2k+1}=$d^+(v_{2k+1})$,*

then this sequence of positive integers is a non-increasing sequence. Moreover, if v_i and v_{i+1} are two consecutive vertices in the sequence such that $d^-(v_{i+1}) \neq d^+(v_i)$, then $s_{i+1} < s_i$.

Using this claim, we can conclude the proof of Theorem 3.3. Let there be one vertex $w \in V$ such that $d^+(u) \neq d^-(w)$ for some edge $(u, w) \in \overrightarrow{E}$. Let w' be the vertex such that $(w', w) \in \overrightarrow{E}$ and

$$d^+(w') = \min\{d^+(u) : (u, w) \in \overrightarrow{E}\}.$$

Since we have already argued that in the graph every connected component has to be strongly connected - there exists a directed path from vertex w to w'. Thus, we can create an infinite sequence of vertices such that w and w' appear consecutively and infinitely often. Now by Claim 3.4, the sequence $\{S\}$ is a non-increasing sequence that decreases infinitely many times. But this cannot happen as all the numbers in the sequence $\{S\}$ represent in-degree or out-degree of vertices and hence, are always finite integers and can never be negative. Thus, if there exists one vertex $w \in V$ such that $d^+(u) \neq d^-(w)$ for some edge $(u, w) \in \overrightarrow{E}$, then we hit a contradiction.

Thus, for all edges $(u, v) \in \overrightarrow{E}$, $d^+(u) = d^-(v)$. □

Proof of Claim 3.4. Since we have assumed that the uniform distribution is a stationary distribution of the Markov chain, then for all vertices v,

$$\sum_{u:(u,v)\in\overrightarrow{E}} \frac{1}{d^+(u)} = 1 = \sum_{w:(v,w)\in\overrightarrow{E}} \frac{1}{d^+(v)}.$$

Let us first prove that in the sequence $\{S\}$, $s_{2i} \geq s_{2i+1}$.
Since $d^+(v_{2i+1}) = \min\left\{d^+(w) \ : \ (w, v_{2i}) \in \overrightarrow{E}\right\}$,

$$1 = \sum_{(u,v_{2i})\in\overrightarrow{E}} \frac{1}{d^+(u)} \leq \frac{d^-(v_{2i})}{d^+(v_{2i+1})}, \tag{3.1}$$

and hence, we have $d^-(v_{2i}) \geq d^+(v_{2i+1})$ which by definition gives $s_{2i} \geq s_{2i+1}$.

Now, let us also prove that in the sequence $\{S\}$, $s_{2i-1} \geq s_{2i}$. By definition, this is same as proving $d^+(v_{2i-1}) \geq d^-(v_{2i})$. Since we have assumed that the graph is a *degree*-Δ graph, proving $s_{2i-1} \geq s_{2i}$ is same as proving $d^-(v_{2i-1}) \leq d^+(v_{2i})$.

Similar to previous case, since

$$d^+(v_{2i}) = \max\left\{d^+(w) \ : \ (w, v_{2i-1}) \in \overrightarrow{E}\right\}$$

and

$$1 = \sum_{(u,v_{2i-1})\in\overrightarrow{E}} \frac{1}{d^+(u)} \geq \frac{d^-(v_{2i-1})}{d^+(v_{2i})}, \tag{3.2}$$

hence, we have $s_{2i-1} \geq s_{2i}$.

Note that the inequalities in Equations 3.1 and 3.2 are strict inequalities if $d^+(v_{2i+1}) \neq d^-(v_{2i})$ and $d^+(v_{2i-1}) \neq d^-(v_{2i})$ respectively. Thus, if for any i, $d^+(v_{i+1}) \neq d^-(v_i)$, then $s_{i+1} < s_i$. □

From Theorem 3.3, we can also obtain the following corollary.

Corollary 3.5. *Let $\overrightarrow{G} = (V, \overrightarrow{E})$ be a connected degree-Δ oriented graph. Then, the uniform distribution on the vertices of $\overrightarrow{G}$ is a stationary distribution of the Markov chain, if and only if the following conditions apply:*

1. *If $G = (V, E)$ is non-bipartite, then the graph $\overrightarrow{G}$ is Eulerian.*
2. *If G is bipartite with bipartition $V_1 \cup V_2 = V$, then $|V_1| = |V_2|$ and in-degree of all vertices in one partition will be same and it will be equal to out-degree of all vertices in the other part.*

Proof. (Part 1) From Theorem 3.3, we follow that the uniform distribution of vertices is a stationary distribution of the Markov chain iff for all $(u, v) \in \overrightarrow{E}$, we have

$$d^+(u) = d^-(v). \tag{3.3}$$

In an Eulerian graph, all vertices will have same in-degree and out-degree (as all vertices in $\overrightarrow{G}$ have the same total degree), hence Condition 3.3 is satisfied. And using Theorem 3.3, we conclude that the uniform distribution is a stationary distribution. Now, if $\overrightarrow{G}$ satisfies Condition 3.3, then if there is an edge between vertex u and v (either $(u, v) \in \overrightarrow{E}$ or $(v, u) \in \overrightarrow{E}$), then $d^-(v) = d^+(u)$. Thus, in an undirected graph, if there is a path of odd length from u to v, then $d^-(v) = d^+(u)$. Thus, if the undirected graph has an odd cycle passing through v, then $d^-(v) = d^+(v)$. Note that, if there exist a vertex having in-degree equal to out-degree and Condition 3.3 holds, then all the vertices in the connected component have in-degree equal to out-degree. This implies that every connected component that has an odd cycle must be Eulerian.

(Part 2) If we start traversing from u (a vertex from left partition of bipartite graph $\overrightarrow{G}$) on underlying undirected graph G of $\overrightarrow{G}$, then all vertices which are at even length distance from u will fall on left partition of $\overrightarrow{G}$ and will have out-degree $d^{+}(u)$. Now proof of Corollary follows by Equation 3.3. $\square$

Theorem 3.3 and Corollary 3.5 have an application to property testing. We present this application in the next section.

3.4 Application to Property Testing

A property of a graph that is invariant under graph isomorphism is called a graph property. Testing of graph properties has been a very hot topic in the area of property testing (see [41], [69]). Note that, whether the uniform distribution on the vertices of $\overrightarrow{G}$ is a stationary distribution of the Markov chain, is actually a graph property. When the input is a graph (as in our case), how to query the input is very important. Various models have been studied in this respect, for example: dense-graph models, sparse-graph models, orientation model etc.

Thus, the problem of distinguishing between whether the stationary distribution of the Markov chain is the uniform distribution or is "far" from it, is a question of testing graph properties. Here, this question has been framed in the *orientation model* (defined in the next subsection). Some interesting graph properties like connectivity [29] and Eulerianity [42] have been studied in this model. Using Theorem 3.1, we show that for both bipartite and non-bipartite graphs, testing (in the orientation model) whether the uniform distribution on vertices of $\overrightarrow{G}$ is the stationary distribution of the Markov chain (generated by a *lazy random walk* on $\overrightarrow{G}$), can be reduced to testing if the graph is Eulerian. Using algorithms from [42] of testing Eulerianity in the orientation model, we obtain various bounds on the query complexity for testing uniformity of the stationary distribution.

In [42], it is shown that if G is an α-expander then whether G is Eulerian can be tested using $O(\Delta/\alpha)$ number of queries in the orientation model. From our result, it implies that if G is an α-expander, then testing uniformity of the stationary distribution can be done with $O(\Delta/\alpha)$ queries. Since $1/\alpha$ is also a measure of the mixing time of the random walk, it implies that the query complexity for testing uniformity of the stationary distribution is directly proportional to the mixing time of the Markov chain.

3.4.1 Property Testing in the Orientation Model

Given an oriented graph $\overrightarrow{G} = (V, \overrightarrow{E})$ and a property $\mathcal{P}$, we want to test whether $\overrightarrow{G}$ satisfies the property or it is "ϵ-far" from satisfying the property. In the orientation model, the underlying graph $G = (V, E)$ is known in advance. Each edge in E is oriented (that is directed in exactly one direction). The orientation of the edges has to be queried. The graph is said to be "ϵ-far" from satisfying the property $\mathcal{P}$ if one has to reorient at least ϵ fraction of the edges to make the graph satisfy the property.

The goal is to design an ϵ-tester (that is, a randomized algorithm) that queries the orientation of the edges and does the following:

- if $\overrightarrow{G}$ satisfies the property $\mathcal{P}$, then the tester ACCEPTS with probability at least $2/3$,
- if $\overrightarrow{G}$ is "ϵ-far" from satisfying the property $\mathcal{P}$, then the tester REJECTS with probability at least $2/3$.

The query complexity of the algorithm is defined by the number of edges it queries. The natural goal is to design a tester for $\mathcal{P}$ with minimum query complexity. If the graph satisfies the property and the tester accepts with probability 1, then the tester is called a 1-*sided error tester*. The usual tester is called a 2-*sided error tester*.

The **orientation model** for testing graph properties was introduced by Halevy *et al.* (see [45]). In [42], Fischer *et al.* studied the problem of testing whether an oriented graph $\overrightarrow{G}$ is Eulerian. They derived various upper and lower bounds for both 1-sided and 2-sided error testers. They even considered various special classes of graphs such as expanders. In this thesis, we use their algorithms for testing whether the uniform distribution is the stationary distribution of the Markov chain.

The reason of choosing the orientation model (not any other graph model like dense, sparse model etc.) for property testing is because in this model, we need to know the underlying undirected graph in advance, which is a regular graph in our case. And we query orientation of edges in order to distinguish whether the input satisfies the property or is far from satisfying it, which in our case is to test whether the stationary distribution is the uniform distribution, or is far from being it.

3.4.2 Testing for Uniformity of Stationary Distribution in the Orientation Model

Given a *degree*-Δ oriented graph $\overrightarrow{G} = (V, \overrightarrow{E})$, we say that the graph has the property $\mathcal{P}'$ if for all $(u, v) \in \overrightarrow{E}$, we have $d^{+}(u) = d^{-}(v)$.

In Theorem 3.3, we proved that the uniform distribution is a stationary distribution of the Markov chain on a *degree*-Δ oriented graph $\overrightarrow{G}$ iff the graph $\overrightarrow{G}$ satisfies the property $\mathcal{P}'$. Thus, given a *degree*-Δ oriented graph $\overrightarrow{G}$, testing whether the uniform distribution is the stationary distribution of the Markov chain (generated by lazy random walk on $\overrightarrow{G}$), is same as testing if the graph has the property $\mathcal{P}'$. Thus, we want to test whether the graph $\overrightarrow{G}$ has the property $\mathcal{P}'$ or is "far" from having the property.

Note that the stationary distribution is uniform for the Markov chain on the whole graph $\overrightarrow{G}$ iff the stationary distribution is uniform for the Markov chain on every connected component of $\overrightarrow{G}$. Since the undirected graph is known in advance, we have the connected components and hence, we need to test each component separately. Also, note that if the graph $\overrightarrow{G}$ is "ϵ-far" from satisfying the property $\mathcal{P}'$, then there is at least one connected component of $\overrightarrow{G}$ that is also "ϵ-far" from satisfying the property $\mathcal{P}'$. Thus, we can do testing connected-component wise and without loss of generality, we can assume that the graph $\overrightarrow{G}$ is connected.

Let us assume $\overrightarrow{G} = (V, \overrightarrow{E})$ is connected. From Corollary 3.5, if $\overrightarrow{G}$ is non-bipartite, then we have to test whether $\overrightarrow{G}$ is Eulerian. Since we can determine whether a connected component is bipartite or not just by looking at the underlying undirected graph. Thus, if $\overrightarrow{G}$ is non-bipartite, then we use the testing algorithm from [42] to test whether the graph is Eulerian.

Now let $\overrightarrow{G}$ be bipartite. Since $\overrightarrow{G}$ is connected, there is a unique partition of the vertex set of $\overrightarrow{G}$ that makes it bipartite and that partition can be found in the preprocessing phase just by looking at the underlying undirected graph. Let the bipartition be V_L and V_R. If $|V_L| \neq |V_R|$, then the graph surely does not have uniform distribution as the stationary distribution for the Markov chain on $\overrightarrow{G}$. By Corollary 3.5, if $|V_L| = |V_R|$, then the graph must have the property that the out-degree of all vertices in V_L must be equal to the in-degree of all vertices in V_R and vice versa.

Let v be a vertex in V_L and $d^-(v) = k_1$ and $d^+(v) = k_2$. Now consider any bipartite directed graph $\overrightarrow{G^*} = (V, \overrightarrow{E^*})$ with bipartition V_L and V_R that satisfies the following conditions:

- The underlying undirected graphs of $\overrightarrow{G}$ and $\overrightarrow{G^*}$ are exactly same,
- For all $v \in V_L$, $d^-_{\overrightarrow{G^*}}(v) = k_2$ and $d^+_{\overrightarrow{G^*}}(v) = k_1$, and
- For all $v \in V_R$, $d^-_{\overrightarrow{G^*}}(v) = k_1$ and $d^+_{\overrightarrow{G^*}}(v) = k_2$.

Note that if such a graph $\overrightarrow{G^*}$ does not exist, then it means that $\overrightarrow{G}$ cannot have the property $\mathcal{P}'$. If such a graph exists, then consider the graph $\overrightarrow{G^{\oplus}} = (V, \overrightarrow{E} \cup \overrightarrow{E^*}) = (V, \overrightarrow{E}^{\oplus})$ obtained by superimposing $\overrightarrow{G}$ and $\overrightarrow{G^*}$- an edge $e \in \overrightarrow{E}^{\oplus}$ if either $e \in \overrightarrow{E}$, or $e \in \overrightarrow{E^*}$. Clearly, if $\overrightarrow{G}$ has the property $\mathcal{P}'$, then $\overrightarrow{G^{\oplus}}$ is Eulerian, and farness from having property $\mathcal{P}'$ is also true by the following lemma:

Lemma 3.6. *If $\overrightarrow{G}$ is "ϵ-far" from having property $\mathcal{P}'$, then $\overrightarrow{G^{\oplus}}$ is "$\frac{\epsilon}{2}$-far" from being Eulerian.*

Proof. Let $\overrightarrow{H}$ be Eulerian graph which is the closest to $\overrightarrow{G^{\oplus}}$. Since the underlying undirected graph for $\overrightarrow{G}$ and $\overrightarrow{G^*}$ are exactly same, there is one-to-one correspondence between the edges in $\overrightarrow{E}$ and $\overrightarrow{E^*}$.

Now, look at edges of $\overrightarrow{G^{\oplus}}$ that were flipped in order to obtain $\overrightarrow{H}$. Suppose a flipped edge (say e) belonging to $\overrightarrow{E^*}$. Then, we can re-flip this edge e and flip the corresponding edge in $\overrightarrow{E}$.

Thus, we have effectively flipped the same number of edges. By performing these operations on the flipped edges of $\overrightarrow{E^*}$, we have obtained a new graph which has the same number of flipped edges as $\overrightarrow{G^\oplus}$ and all the flipped edges in $\overrightarrow{G^\oplus}$ belong to $\overrightarrow{E}$.

Thus, if the graph $\overrightarrow{G^\oplus}$ is not "$\frac{\epsilon}{2}$-far" from being Eulerian then $\overrightarrow{G}$ is not "ϵ-far" from having property $\mathcal{P}'$, which is a contradiction. $\square$

Now, all we have to test is whether the new graph $\overrightarrow{G^\oplus}$ is Eulerian or "$\frac{\epsilon}{2}$-far" from being Eulerian. Note that every query to $\overrightarrow{G^\oplus}$ can be simulated by a single query to G. Thus, we can now use the Eulerian testing algorithm from [42]. The algorithm is summarized in Algorithm 5, and the various bounds on the query complexity that can be obtained is summarized in Table 3.1 (see next page).

```
Data: Degree-$\Delta$ Oriented Graph $\overrightarrow{G} = (V, \overrightarrow{E})$
Result: Whether $\overrightarrow{G}$ has the property $\mathcal{P}'$ or is "$\epsilon$-far" from having it.
1  if $G$ is non-bipartite then
2      Test if $\overrightarrow{G}$ is Eulerian (see [42]) and give the corresponding output.
3  else
4      Let $V_L$ and $V_R$ be the bipartition for the graph $G$.
5      Sample a vertex from $V_L$ and query all edges incident to it. Let $d^-(v) = k_1$ and
       $d^+(v) = k_2$.
6      Construct any bipartite graph $\overrightarrow{G^*} = (V, \overrightarrow{E^*})$ with bipartition $(V_L, V_R)$ such that
7      (a) For all $v \in V_L$, $d^-_{\overrightarrow{G^*}}(v) = k_2$ and $d^+_{\overrightarrow{G^*}}(v) = k_1$, and for all $v \in V_R$, $d^-_{\overrightarrow{G^*}}(v) = k_1$
       and $d^+_{\overrightarrow{G^*}}(v) = k_2$.
8      (b) The underlying graph of $\overrightarrow{G^*}$ is exactly same as $G = (V, E)$.
9      Superimpose $\overrightarrow{G^*}$ and the graph $G$ (say $\overrightarrow{G^\oplus} = (V, E \cup \overrightarrow{E^*})$).
10     Test if $\overrightarrow{G^\oplus}$ is Eulerian (see [42]) and give the corresponding output.
11 end
```

Algorithm 5: Algorithm for testing property $\mathcal{P}'$

3.5 Conclusion and Open Problems

We have shown that, for a given *degree*-Δ oriented graph $\overrightarrow{G} = (V, \overrightarrow{E})$, whether the uniform distribution on vertices of $\overrightarrow{G}$ is a stationary distribution of the Markov chain, depends on a local property of graph. If $\overrightarrow{G}$ satisfies this local property then it has some particular kind of structure (see Corollary 3.5). Finally, as an application of this result, we showed that testing this local property in orientation model, can be reduced to testing Eulerianity (see [42]).

It is an interesting problem to test whether the stationary distribution of a Markov chain is close to some fixed distribution D. In this thesis, we have considered D to be uniform

	1-sided test	**2-sided test**
Graphs with large Δ	$\Delta + O(\frac{m}{\epsilon^2\Delta})$	$\Delta + \min\left\{\tilde{O}\left(\frac{m^3}{\epsilon^6\Delta^6}\right), \tilde{O}\left(\frac{m}{\epsilon^2\Delta^{\frac{3}{2}}}\right)\right\}$
Bounded-degree graphs *	$\Omega\left(m^{\frac{1}{4}}\right)$	$\Omega\left(\sqrt{\frac{\log m}{\log\log m}}\right)$
α-expander	$O\left(\frac{\Delta\log\left(\frac{1}{\epsilon}\right)}{\alpha\epsilon}\right)$	$min\left\{\tilde{O}\left(\left(\frac{\log\left(\frac{1}{\epsilon}\right)}{\alpha\epsilon}\right)^3\right), \tilde{O}\left(\left(\frac{\sqrt{\Delta}\log\left(\frac{1}{\epsilon}\right)}{\alpha\epsilon}\right)\right)\right\}$

* Lower bound holds for 4-regular graph.
Δ is the maximum degree of the underlying undirected graph and m is the number of edges is the graph.

Table 3.1: Bounds on the query complexity (in the orientation model) for testing uniformity of stationary distribution of the Markov chain obtained by the random walk on a directed graph.

distribution, but the same problem is also interesting for other distributions. Moreover, finding a relationship between the distance of the stationary distribution from the uniform distribution, and the number of edges that needs to be reoriented, is an interesting open problem. Also, this result holds only for graphs where the in-degree plus out-degree of all the vertices are same. A major open problem of this thesis is to come up with a similar statement for more general graphs.

Chapter 4

Computing Bits of Algebraic Numbers

4.1 Introduction

Algebraic numbers are (real or complex) roots of finite degree polynomials with integer coefficients. Needless to say, they are fundamental objects and play an important part in all the branches of Mathematics. The equivalence between reals with recurring binary expansions (or expansions in any positive integral radix) and rationals is easy to observe. Thus, computing the bits of fixed rationals is computationally uninteresting. However, the problem becomes interesting if we focus on irrational real numbers. For example: how hard/easy is it to compute 10^{100}-th bit of $\sqrt{2}$? Computability of such numbers heralded the birth of of Computer Science in Turing's landmark paper [77] where the computability of the digits of irrationals like π and e is first addressed.

Building on the surprising Bailey-Borwein-Plouffe (BBP) formula [17] for π, Yap [80] shows that certain transcendental numbers such as π have binary expansions computable in a small complexity class like deterministic logarithmic space. Motivated by this result, we seek to answer the corresponding question for algebraic numbers. The answers we get turn out to be unsatisfactory but intriguing in many respects. In a nutshell, we are able to show only a very weak upper bound to the "succinct" version of the problem and virtually no lower bounds. This gap between best known (at least to our knowledge) upper and lower bounds easily beats other old hard-to-crack chestnuts such as graph isomorphism and integer factorization.

4.1.1 Versions of the Problem

The problem as stated in [80] asks for the n-th bit of the (infinite) binary sequence of an irrational real given n in *unary*. This version of the problem is called the "verbose" version. The "succinct" version of the problem asks for the n-th bit given n in *binary*. Obviously, the suc-

This chapter reports about our work done in [38].

cinct version of the problem is naturally much harder than the corresponding verbose version. We can solve the verbose version in GapNC^1, a subclass of logspace. For the succinct version of the problem we are unable to prove a deterministic polynomial or even a non-deterministic polynomial upper bound. The best we can do is place it at a finite level in the counting hierarchy (which includes the computation of the permanent at its first level). Even more surprising is that we are unable to prove any non-trivial lower bound for any irrational algebraic number. Intriguingly, we can prove Parity and in general $\mathsf{AC}^0[p]$ lower bounds for computing specific *rationals*.

4.1.2 Previous Proof Techniques

In his article [80], Yap used a BBP like [16] series to prove a logspace upper bound for the (verbose version of) computing the bits of π. At the core of that argument is the concept of bounded irrationality measure of π which intuitively measures how inapproximable π is, by rationals. Roughly, the BBP-like series was used to approximate π by rationals and then argue, via the bounded irrationality measure, that, since there aren't too many good approximations to π, the computed one must match the actual expansion to lots of bit positions.

4.1.3 Our proof technique

Further progress was stymied by the extant ignorance of BBP like series for most well-known irrationals. Our crucial observation is that approximating an irrational can be accomplished by means other than a BBP-like series for instance by using Newton-Raphson. Bounded irrationality measure for algebraic numbers follows by a deep theorem of Roth [71]. But we show that we can keep our proof elementary by replacing the Roth's theorem by the Liouville's Theorem [46] which has a simple and elementary proof (see e.g. [75]).

An upper bound on the verbose version of the problem can be obtained by observing iterations of Newton Raphson, which can be viewed as composition of polynomials, which in turn can expressed in the form of an arithmetic circuit. Each layer in the circuit corresponds to an iteration of Newton Raphson. Since Newton Raphson converges quickly, we show that the size of the circuit would be at most polynomial. Thus, it implies that computing bits of algebraic numbers boils down to computing bits of polynomial size arithmetic circuits.

An upper bound on the succinct version of the problem follows by observing that Newton Raphson can be viewed as approximating the algebraic number by a rational which is the ratio of two *Straight Line Programs* or *SLP*'s. Thus, the problem of computing bits of algebraic numbers boils down to the problem of computing bits of the ratio of two polynomial size SLP's. This problem has been studied in literature under the name of BitSLP [5, 4]. In [4], it is shown that how to place BitSLP in a finite level of the counting hierarchy, and we could simply use their upper bound for our result.

4.1.4 Our Results

Yap [80] showed that the bits of π are in Logspace. This was the origin of this endeavour and we are able to refine his result to the following:

Theorem 4.1. *Let α be a real number with bounded irrationality measure, which can be expressed as a convergent series and further the the m^{th} term (for input m in unary) is in* FTC^0*. Then, computing the n^{th} bit of α is in* TC^0*, if n is given in unary, and is reducible to* BitSLP[†]*, if n is given in binary.*

In particular, the above inclusions hold for π.

Somewhat paradoxically, we get slightly weaker bounds for algebraic numbers. As our main result, we are able to show the following theorem, which is a restatement of Theorem 1.6 (for an explanation of the complexity classes used in the statement please see the next section):

Theorem 4.2. *Let p be a fixed univariate polynomial of degree d, having integer coefficients. If n is given in unary, then every real roots of p can be computed in the function class* GapNC^1*, and the n^{th} bit of every real root can be computed in* TCLL*. Also, if n is given in binary, then the problem of computing the n^{th} bit of each real root of p is reducible to* BitSLP[†].

4.1.5 Related Work

Jerábek [53] also considered the problem of computing the n^{th} bit of an algebraic number. He showed that for any constant d, all real and complex roots of any degree d univariate rational polynomial (given by a list of coefficients in binary) can be computed to a given accuracy by a uniform TC^0circuit. The main idea behind his approach is to compute the inverse function of the polynomial by a power series. There are two main differences between our result and the result from [53]: we solve for real roots only while he solves for complex roots also; he considers only verbose version of the problem (whose upper bound is better than ours as, $\mathsf{TC}^0 \subseteq \mathsf{GapNC}^1$) while we solve the problem for the succinct version also.

A similar problem has been studied by Ben-Or, Feig, Kozen and Tiwari [27]. Given a polynomial of degree d with m bit integer coefficients and an integer μ (where μ is specified in unary), they consider the problem of determining all its roots, under the promise that *all roots are real*. Under this assumption, they are able to show that approximating the roots to an additive error of $2^{-\mu}$ is in NC. Notice that while their result concerns non-constant algebraic numbers, it does not involve finding the bits of algebraic numbers but only approximating them. Also, since they only achieve unary tolerance, their claimed upper bound of NC is not better

[†]It was wrongly quoted as $\mathsf{PH}^{\mathsf{PP}^{\mathsf{PP}}}$ in our paper [38]. Allender *et al.* [4] showed that $\mathsf{BitSLP} \in \mathsf{PH}^{\mathsf{PP}^{\mathsf{PP}^{\mathsf{PP}}}}$, we can use their upper bound for our case.

than $\mathsf{GapNC}^1 \subseteq \mathsf{NC}$. Moreover, their method does not work if the all-real-roots promise is not satisfied.

Reif and Neff [65] considered the problem of approximating (complex) roots of complex polynomials. Given a univariate polynomial of degree d with complex coefficients (with norms less than 2^m in magnitude), they considered a problem of finding all the roots of the polynomial up to specified precision of $2^{-\mu}$. Although their result concerns non-constant complex algebraic numbers, it does not compute the bits of algebraic number but only approximates them. If μ is specified in unary, then in the arithmetic model for computation, both of their algorithms (sequential and parallel) of approximating roots of a constant polynomial takes logarithmic running time (logarithmic in length of μ). Their algorithms are designed for a different model of computation (arithmetic model of computation), so it is incomparable with our result.

4.1.6 Organization of the chapter

In Section 4.2, we start with pointers to relevant complexity classes, and more importantly known results from elementary analysis that we will need in our proofs. In Section 4.3, we provide upper bounds on the complexity of composing bivariate polynomials. This is used in the subsequent Section 4.4, where we view Newton-Raphson as an iterated composition of bivariate polynomials. In this section, we prove that the method converges "quickly" if its initial point is in a carefully picked interval and that we can efficiently identify such intervals. In Section 4.5, we make use of the tools we have put together in the previous sections to prove theorems 4.1, 4.2. We also prove a lower bound on rationals in this section. Finally in Section 4.6, we conclude with some open questions.

4.2 Preliminaries

4.2.1 Complexity Theoretic Preliminaries

We start off by introducing straight line programs. An arithmetic circuit is a directed acyclic graph with input nodes labeled with the constants $0, 1$ or with indeterminates $X_1, \ldots, X_k$ for some k. Internal nodes are labeled with one of the operations $+, -, *$. A *straight-line program* is a sequence of instructions corresponding to a sequential evaluation of an arithmetic circuit. We will need to refer to standard complexity classes like NP, PP, and PH and we refer the reader to any standard text in complexity such as [11]. We will also use circuit complexity classes like TC^0 and GapNC^1 and we refer the reader to [78] for details.

One non-standard class we use is TCLL. This is inspired by the class FOLL introduced in [18] which is essentially the class of languages accepted by a uniform version of an FAC^0 circuit iterated $O(\log\log n)$ many times with an AC^0-circuit on top. We obtain a TCLL-circuit by adding a TC^0-circuit on top of the iterated block of FAC^0-circuits. The class of languages accepted by such circuits constitutes TCLL.

4.2.2 Mathematical Preliminaries

In order to upper bound the largest magnitude of roots of a polynomial we note the following (e.g. see Section 2 of Chapter 6 of [79]):

Fact 4.3. *(Cauchy) Let $p(x) = \sum_{i=0}^{d} a_i x^i$ be a polynomial. Then, every root of $p(x)$ is smaller in absolute value than:*

$$M_p = 1 + \frac{1}{|a_d|}\max_{i=0}^{d-1}|a_i|$$

We can consider $[-M_p, M_p]$ as the possible solution range which contains all the real roots.

Fact 4.4. *The Taylor (see [1]) series of a real (complex) function $f(x)$ that is infinitely differentiable in a neighborhood of a real (complex) number a is the power series*

$$\sum_{n=0}^{\infty} \frac{f^{(n)}(a)}{n!}(x-a)^n$$

where $f^{(n)}(a)$ denotes the n^{th} derivative of f evaluated at the point a.

We will need to lower bound the minimum distance between the roots of a polynomial, or the so called *root separation*. We use the following version of the Davenport-Mahler theorem (see e.g. Corollary 29 (i) of Lecture VI from Yap [79] for details of notation and proof):

Fact 4.5. *(Davenport-Mahler) The separation between the roots of a univariate polynomial $p(x)$ of degree d is at least:*

$$\sqrt{3|disc(p)|}||p||_2^{-d+1}d^{-(d+2)/2}$$

Here, $disc(p)$ is the *discriminant* of p and $||p||_2$ is the 2-norm of the coefficients of p. Further notice that a lower bound approximation to this bound can be computed in FTC^0 for constant polynomials p (since it involves computing lower bound *approximations* for radicals and powers of constants and constant determinants).

We will also need an upper bound on the magnitude of the derivative of a univariate polynomial in the open interval (a, b). This is given by the so called the Markoff's Theorem [60] (see also [66]):

Fact 4.6. *Let $p(x)$ be a degree d polynomial satisfying,*

$$\forall x \in (a,b), |p(x)| \leq M_{(a,b)}$$

Then, the derivative $p'(x)$ satisfies:

$$\forall x \in (a,b), |p'(x)| \leq M'_{(a,b)} = \frac{d^2 M_{(a,b)}}{b-a}$$

Removing Rational and Repeated Roots

The rational roots of a polynomial can be dealt with, by using the following, easy to see, fact:

Fact 4.7. *The rational roots of a monic polynomial (i.e., the highest degree coefficient is* 1*) with integer coefficients are integers.*

To make $p(x) = \sum_{i=0}^{d} a_i x^i$ monic, just substitute $y = a_d x$ in $a_d^{d-1} p(x)$ to obtain a monic polynomial $q(y)$. Iterating over all integers in the range given by Fact 4.3, we can find and eliminate the integer roots of $q(y)$, and therefore, the corresponding rational roots of $p(x)$.

In general, a polynomial will have repeated roots. If this is the case then, the repeated root will also be a root of the derivative. Thus, it suffices to find the gcd g of the given polynomial p and its derivative p', since we can recursively find the roots of p/g and g. As we will be focusing only on fixed polynomials the gcd computation and the division will be in FTC^0.

We will actually need a more stringent condition on the polynomials - *i.e.*, p does not share a root with its derivative p' and even with its double derivative p''. Notice that the above procedure does not guarantee this. For example, if $p(x) = x(x^2-1)$, then $p'(x) = 3x^2 - 1$ and $p''(x) = 6x$. Thus, p and p'' share a root but p and p' don't.

So, we will follow an iterative procedure in which we find the gcd of p, p' to get two polynomials $p_1 = p/(p,p'), p_2 = (p,p')$ (denoting gcd of p, q by (p,q)). For p_1 we find the gcd (p_1, p_1'') and set $p_3 = p_1/(p_1, p_1''), p_4 = (p_1, p_1'')$. Now, we recurse for the 3 polynomials p_2, p_3, p_4 (whose product is p). Notice that the recursion will bottom out when some gcd becomes a constant. As a result of the above discussion, we can assume hereafter that p does not share a root with either p' or p''.

Good Intervals

Definition 4.8. *Fix an interval $I = [a,b]$ of length $|I| = b - a$. We will call the interval I good for an integral polynomial p if it contains exactly one root (say α) of p and no root of p' and p''.*

Lemma 4.9. *If p is a polynomial of degree d such that p doesn't share a root with its derivative and double derivative, then there exist δ (depending only on the polynomial p) such that all intervals of length less than δ contain at most one root of p, and if they contain a root of p, then they do not contain any roots of p' or p''. As a consequence, we can fix good intervals I.*

Proof. Using Fact 4.5 compute δ, *i.e.*, the minimum distance between roots of p and that of $p'p''$. (let p_1, p_2 be p', p'' with the roots common with p'', p''' respectively, removed; then a lower bound on the root separation of pp_1p_2, does the job). Partition the interval containing all the roots (which can be inferred from Fact 4.3) into sub-intervals of length δ. If such an interval contains a root of p, then it contains precisely one root of p and no roots of p' or p''. By checking that signs of p are opposite at the end points we can fix good intervals. □

4.2.3 From approximation to exact computation

We will crucially use the following theorem (see e.g. Shidlovskii[75] or Yap [79])

Fact 4.10. *(Liouville's Theorem) If x is a real algebraic number of degree $d \geq 1$, then there exists a constant $c = c(x) > 0$ such that the following inequality holds for any $\alpha \in \mathbb{Z}$ and $\beta \in \mathbb{N}$, $\alpha/\beta \neq x$:*

$$\left| x - \frac{\alpha}{\beta} \right| > \frac{c}{\beta^d}$$

Roth's theorem sharpen the inequality of the Liouvilles theorem, we formally state the theorem as follows:

Fact 4.11. *(Roth's Theorem [71]) If x is a real algebraic number of degree $d \geq 1$, then there exists a constant $c = c(x, \epsilon) > 0$ such that the following inequality holds for any $\alpha \in \mathbb{Z}$ and $\beta \in \mathbb{N}$, $\alpha/\beta \neq x$:*

$$\left| x - \frac{\alpha}{\beta} \right| > \frac{c(x, \epsilon)}{\beta^{2+\epsilon}}$$

This theorem is best possible of its kind; the number 2 in the exponent cannot be decreased.

The rest of this subsection is an adaptation of the corresponding material in Chee Yap's paper on computing π in L. The primary difference being that we choose to pick the elementary Liouville's Theorem for algebraic numbers instead of the advanced arguments required for bounding the irrationality measure of π. We could throughout replace the use of Liouville's theorem by the much stronger and deeper Roth's theorem but prefer not to do so in order to retain the elementary nature of the arguments.

Definition 4.12. *Let x be a real number. Let $\{x\} = x - \lfloor x \rfloor$ be the fractional part of x. Further, let $\{x\}_n = \{2^n x\}$ and x_n be the n-th bit after the binary point.*

It is clear that $x_n = 1$ iff $\{x\}_{n-1} \geq \frac{1}{2}$. For algebraic numbers we can sharpen this:

Lemma 4.13. *(Adapted from Yap[80]) Let x be an irrational algebraic number of degree d, and let $c = c(x)$ be the constant guaranteed by Liouville's theorem. Let $\epsilon_n = c2^{-(d-1)n-2}$, then for n such that $\epsilon_n < \frac{1}{4}$ we have:*

- $x_n = 1$ *iff* $\{x\}_{n-1} \in (\frac{1}{2} + 2\epsilon_n, 1 - 2\epsilon_n)$.
- $x_n = 0$ *iff* $\{x\}_{n-1} \in (2\epsilon_n, \frac{1}{2} - 2\epsilon_n)$.

Proof. Taking $\beta = 2^n$ in Liouville's theorem we get: $|x - 2^{-n}\alpha| > \frac{c(x)}{2^{dn}}$ i.e., $\left|2^{n-1}x - \frac{\alpha}{2}\right| > \frac{c(x)}{2^{d(n-1)+1}} = 2\epsilon_n$. $\square$

Consequently, we can find successive approximations $\{S_m\}_{m\in\mathbb{N}}$ such that the error terms $R_m = x - S_m$ are small enough . x_n is just the n-th bit of S_m.

4.3 Complexity of Composition

We investigate the complexity of composing polynomials. This will be useful when we use the Newton-Raphson method to approximate roots of polynomials since Newton-Raphson can be viewed roughly as an algorithm that iteratively composes polynomials.

Definition 4.14. *A univariate polynomial with integer coefficients is, an* integral *polynomial. Any integral polynomial when evaluated on a rational value $\frac{\alpha}{\beta}$, where α, β are integers, can be expressed as the ratio of two bivariate polynomials in α, β called the* ratio *polynomials of the integral polynomial.*

Definition 4.15. *Let p be an integral polynomial. For a positive integer t, the t-*composition *of the polynomial, denoted by $p^{[t]}$, is defined inductively as: $p^{[1]}(x) = p(x)$ and $p^{[t+1]}(x) = p^{[t]}(p(x))$.*

Definition 4.16. *Let f, g be a pair of bivariate polynomials. For a positive integer t define the t-bicomposition of (f, g) to be the pair of bivariate polynomials $(F^{[t]}, G^{[t]})$ as follows:*

$$F^{[1]}(\alpha, \beta) = f(\alpha, \beta), G^{[1]}(\alpha, \beta) = g(\alpha, \beta),$$

and,

$$F^{[t+1]}(\alpha, \beta) = f(F^{[t]}(\alpha, \beta), G^{[t]}(\alpha, \beta)),$$
$$G^{[t+1]}(\alpha, \beta) = g(F^{[t]}(\alpha, \beta), G^{[t]}(\alpha, \beta)),$$

The following is a direct consequence of the definitions:

Proposition 4.17. *The ratio polynomials of the t-composition of an integral polynomial p are exactly the t-bicompositions of the ratio polynomials of p.*

Definition 4.18. *For an arithmetic complexity class $\mathcal{C}$ containing FTC^0, we define a bivariate polynomial as $\mathcal{C}$-computable if we can evaluate its value on constant arguments (x, y) in $\mathcal{C}$. Also, we call an integral polynomial $\mathcal{C}$-computable if its ratio polynomials are in $\mathcal{C}$.*

In the above definition, we are referring to bivariate polynomials with constant arguments but non-constant coefficients, though we will evaluate them only on the points independent of the queried input bit. Here, we consider upper bounds on the complexity of computing compositions of fixed integral polynomials. We first prove that:

Lemma 4.19. *Let p be a fixed integral polynomial, then given n in unary the l-bicomposition (for $l = O(\lceil \log n \rceil)$) of p is computable in* GapNC^1.

Proof. From Definition 4.18 and Proposition 4.17, it suffices to prove that the bicomposition of the ratio polynomials of p is computable in GapNC^1. Given a rational number as the bits of its numerator and denominator, we can first obtain the arithmetic values of its numerator and denominator in GapNC^1. Now, we are done modulo the following claim.

Claim 4.20. *Let f be a fixed bivariate polynomial with integral coefficients and let α, β be two integers, then there is a constant depth arithmetic circuit that takes α, β as inputs and outputs $f(\alpha, \beta)$.*

Proof of Claim. The depth of the circuit is seen to be bounded by $O(\lceil \log d \rceil)$, where d is the degree of the polynomial - we just need to find $\alpha^i \beta^j$ by a tree of height $max(\lceil \log i \rceil, \lceil \log j \rceil)+1$, multiply it with the coefficient and add up the results by a tree of depth $\lceil \log{(d+1)} \rceil$. Since degree is a constant we are done. $\square$

$\square$

We now prove an orthogonal bound on the complexity of compositions. But before that we need a small lemma:

Lemma 4.21. *Suppose G is a layered graph of width $O(n)$ and depth $O(\log n)$, then reachability (from a vertex in the first layer to a vertex in the last layer) can be done by an* AC*-circuit of depth $O(\log \log n)$.*

Proof. It suffices to show that the reachability between two layers which are separated by another layer is in AC^0, which is clear. $\square$

Note 4.22. *In fact, from the proof it is clear that, if G contains $O(\log n)$ identical layers, then this reachability is in the class* FOLL *defined in [18].*

Lemma 4.23. *Let p be a fixed integral polynomial, then given n in unary the l-bicomposition (for $l = O(\lceil \log n \rceil)$) of p is computable in* TCLL.

Proof. Notice that if α, β are some fixed integers, the value of the l-bicomposition of the ratio functions on α, β is bounded by an $n^{O(1)}$ bit integer. It suffices to compute the value of this composition modulo all $O(\log n)$ bit primes, since we can do Chinese Remaindering in TC^0 [50]. Fix an $O(\log n)$ bit prime q, and construct the following bipartite graph H_q on vertices S, T (where $|S| = |T| = q^2$) and both S, T consist of pairs ab, $a, b \in \{0, \dots, q-1\}$. $(ab, a'b')$ is an edge iff $f(a,b) \equiv a' \bmod q$ and $g(a,b) = b' \bmod q$, where f, g are the ratio functions of p. Further, let G_q be the layered graph obtained by taking l layers of H_q. Then, it is clear that reachability in G_q from the first to the last layer is exactly equivalent to the values of the l-compositions modulo q. We are done with the aid of Lemma 4.21 and [50]. $\square$

The following is a consequence of the definitions and of [50].

Lemma 4.24. *If p is an integral polynomial which is $\mathcal{C}$-computable ($\mathsf{FTC}^0 \subseteq \mathcal{C}$), then on input m, n in unary, where $m > n$, we can obtain the n-th bit of some number that differs from $p(\frac{\alpha}{\beta})$, by at most 2^{-m} in $\mathcal{C}$.*

Now we describe the binary analog of the above lemma.

Note 4.25. *In the remaining part of this section we denote polynomially bounded* integers *by lower case letters e.g. n, t. We denote those with polynomial number of bits by uppercase letters e.g. N, T. Finally we denote those with exponentially many bits by calligraphic letters e.g. $\mathcal{N}, \mathcal{D}$. This notation does not apply to rationals like u, σ.*

Lemma 4.26. *Let $\mathcal{N}$ and $\mathcal{D}$ be the outputs of two* SLP*'s. Then, the problem of computing the N^{th} (where N is input in binary) bit of an approximation (accurate up to an additive error of $2^{-(N+1)}$) is reducible to* BitSLP.

Proof. We will compute an under approximation of $\frac{\mathcal{N}}{\mathcal{D}}$ with error less than $2^{-(N+1)}$.

Let $u = 1 - \mathcal{D}2^{-T}$ where $T \geq 2$ is an integer such that $2^{T-1} \leq \mathcal{D} < 2^T$. Hence $|u| \leq \frac{1}{2}$.

Notice that the higher order bit of T can be found by using PosSLP : we just need to find an integer t such that $2^{2^t} \leq \mathcal{D} < 2^{2^{t+1}}$ and both these questions are PosSLP questions. Having found T_i a lower bound of T correct up to the higher order i bits of T, i.e. $2^{T_i} \leq \mathcal{D} < 2^{T_i + 2^i}$, we check if $2^{T_i + 2^{i-1}} \leq \mathcal{D}$ and update T_{i-1} to $T_i + 2^{i-1}$ iff the inequality holds (and $T_{i-1} = T_i$ otherwise). Thus by asking a polynomial number of PosSLP queries, we can determine T (each bit of T is in $\mathsf{PH}^{\mathsf{PP}^{\mathsf{PP}}}$).

Now consider the series

$$\mathcal{D}^{-1} = 2^{-T}(1-u)^{-1} = 2^{-T}(1 + u + u^2 + ...)$$

Set $\mathcal{D}' = 2^{-T}(1 + u + u^2 + ...u^{N+1})$, then

$$\mathcal{D}^{-1} - \mathcal{D}' \leq \ 2^{-T} \sum_{I > N+1} 2^{-I} < \ 2^{-(N+1)}$$

Now we need to compute N^{th} bit of

$$\frac{\mathcal{N}}{2^T} \sum_{I=0}^{N+1} (1 - \frac{\mathcal{D}}{2^T})^I = \frac{1}{2^{(N+2)T}} \sum_{I=0}^{N+1} \mathcal{N}(2^T - \mathcal{D})^I 2^{(N+1-I)T} \tag{4.1}$$

We need to compute the $M = N + (N+2)T^{th}$ bit of: $\mathcal{Y} = \sum_{I=0}^{N+1} \mathcal{N}(2^T - \mathcal{D})^I 2^{(N+1-I)T}$. Since each term in the summation is large and there are exponentially many terms in summation, so we will do the computation modulo small primes. Let $\mathcal{Y}_I$ denote the I^{th} term of the summation.

Let $\mathcal{M}_n$ be the product of all odd primes less than 2^{n^2}. For such primes P let $H_{P,n}$ denote inverse of $\frac{\mathcal{M}_n}{P} \bmod P$. Any integer $0 \leq Y_I < \mathcal{M}_n$ can be represented uniquely as a list $(Y_{I,P})$, where P runs over the odd primes bounded by 2^{n^2} and $Y_{I,P} = \mathcal{Y}_I \bmod P$.

Define the family of approximation functions $app_n(\mathcal{Y})$ to be $\sum_P \sum_I Y_{I,P} H_{P,n} \sigma_{P,n}$ where $\sigma_{P,n}$ is the result of truncating the binary expansion of $\frac{1}{P}$ after 2^{n^4} bits. Notice that for sufficiently large n, and $\mathcal{Y} < \mathcal{M}_n$, $app_n(\mathcal{Y})$ is within $2^{-2^{n^3}} < 2^{-(N+1)}$ of $\mathcal{Y}/\mathcal{M}_n$ as in the proof of Theorem 4.2 of [5]. Continuing to emulate that proof further and using the Maciel-Therien (see [59]) circuit for iterated addition, we get the same bound as for BitSLP. Notice that we have a double summation instead of a single one in [5], yet it can be written out as a large summation and thus does not increase the depth of the circuit. □

Corollary 4.27. *Let $\mathcal{N}$ and $\mathcal{D}$ be the outputs of two* SLP*'s. Computing the N^{th} (where N is input in binary) bit of an approximation (accurate up to an additive error of $2^{-(N+1)}$) of $\frac{\mathcal{N}}{\mathcal{D}}$ is in $\mathsf{PH}^{\mathsf{PP}^{\mathsf{PP}^{\mathsf{PP}}}}$.*

Proof. This is immediate from Lemma 4.26 and Corollary 3 of [4]. □

4.4 Establishing Quadratic Convergence

We use the famous Newton-Raphson method to approximate Algebraic Numbers. The treatment is tailored with our particular application in mind. There are some features in the proof (for instance a careful use of Markoff's result on lower bounding the derivative of a polynomial) which led us to prove the correctness and rate of convergence of the method from scratch rather than import it as a black-box.

Definition 4.28. *(Newton-Raphson) Given an integral polynomial p and a starting point x_0, recursively define:*

$$x_{i+1} = x_i - \frac{p(x_i)}{p'(x_i)},$$

whenever x_i is defined and $p'(x_i)$ is non-zero.

Recall good intervals from Definition 4.8.

Definition 4.29. *Given a good interval I containing exactly one root (say α) of an integral polynomial p. Let ϵ_i denote the error in the i^{th} iteration of Newton-Raphson, i.e., $\epsilon_i = |x_i - \alpha|$ when starting with $x_0 \in I$. Notice that ϵ_i is defined only when x_i is.*

Definition 4.30. *We say that Newton-Raphson converges quadratically (with parameter M) for an integral polynomial p whenever M is a non-negative real such that for any interval I which is good for p and of length at most $\min(\frac{1}{4M^2}, \frac{1}{4})$, it is the case that the errors at consecutive iterations (whenever both are defined) satisfy $\epsilon_{i+1} \leq M\epsilon_i^2$.*

The following Lemma shows that not only are the errors at all iterations defined under the assumptions of Lemma 4.30 but also, that, Newton-Raphson converges "quickly".

Lemma 4.31. *If Newton-Raphson converges quadratically (with parameter M) for an integral polynomial p, then for every $i \geq 0$, the i^{th} iterand, x_i, is at distance at most* $\min(\frac{1}{4M^2}, 2^{-2^{i/2}})$ *from the unique root of p in any good interval I of length $|I| \leq \min(\frac{1}{4}, \frac{1}{4M^2})$. In particular, $x_i \in I$ for every $i \geq 0$.*

Proof. We proceed by induction on the number of iterations. For the base case, notice that x_0 is at distance at most $\epsilon_0 \leq \min(\frac{1}{4}, \frac{1}{4M^2})$ from the root.

Now assume that $\epsilon_i < \min(\frac{1}{4M^2}, 2^{-2^{i/2}})$. Then,

$$\begin{aligned}
\epsilon_{i+1} &\leq M\epsilon_i^2 \\
&= (M\sqrt{\epsilon_i})\epsilon_i^{1.5} \\
&\leq (M\sqrt{\frac{1}{4M^2}})\epsilon_i^{1.5} \\
&= \frac{1}{2}\epsilon_i^{1.5} \\
&\leq 2^{-1}2^{-1.5\times 2^{i/2}} \\
&< 2^{-\sqrt{2}\times 2^{i/2}} \\
&= 2^{-2^{(i+1)/2}}
\end{aligned}$$

Since $\epsilon_{i+1} \leq \frac{1}{2}\epsilon_i^{1.5}$ and $\epsilon_i^{0.5} < \frac{1}{2^{2^{(i-1)/2}}} < 1$ for $i \geq 0$, therefore, $\epsilon_{i+1} < \epsilon_i < \frac{1}{4M^2}$ where the second inequality follows from the inductive assumption. This completes the proof of the inductive step. □

Lemma 4.32. *For any integral polynomial p and any good interval I thereof, there exists a subinterval $I' \subseteq I$ such that Newton-Raphson converges quadratically in I'.*

Proof. Let the unique root of the integral polynomial p, contained in I, be α. Thus, $p(\alpha) = 0$. By Definition 4.29, the error in the $i + 1^{th}$ iteration of Newton-Raphson (whenever defined) is:

$$\epsilon_{i+1} = |x_{i+1} - \alpha| = \left|\frac{1}{2}\frac{p''(\xi_i)}{p'(x_i)}\right|\epsilon_i^2$$

Since p' does not have a root in that interval and p'' is finite (because p is a polynomial), the right hand side is well-defined.

In the good interval I, p' is monotonic and hence the minimum (and maximum) value of p' is attained at the end-points of I. Now, we can upper bound the absolute value $|p''|$ using an upper bound for p' and Markoff's result (Fact 4.6). Let this value be denoted by ρ_1 and the minimum value of $|p'|$ by $\rho_2 \neq 0$ ($\rho_2 = 0$ would contradict the assumption that I does not contain a root of p'). Now, set M to be $\frac{\rho_1}{2\rho_2}$.

Partition I into sub-intervals of length $\frac{1}{4M^2}$ and let I' be the unique subinterval containing a root of p : *i.e.* the unique sub-interval such that p takes oppositely signed values at its end points. It is easy to see Newton-Raphson converges quadratically (with parameter M) in I'. $\square$

4.5 Putting it all together

We now complete the proofs of Theorem 4.2, and Theorem 4.1.

Proof of Theorem 4.2. From Lemma 4.9, we can fix a good interval I as fixing good interval doesn't require the knowledge about the queried bit. Now, using Lemma 4.32, we can find a subinterval of I such that Newton-Raphson will converge quadratically in this interval.

Since Newton-Raphson converges quadratically, in order to obtain an inverse exponential error in terms of n, (By Lemma 4.31) we need $O(\lceil \log n \rceil)$ iterations. Now, by Lemma 4.19 and 4.23, along with Lemma 4.24, we get the $O(\lceil \log n \rceil)$ compositions of Newton-Raphson (taking as initial point, the middle point of the interval I' obtained from Lemma 4.32). Thus, the real roots of the polynomial p can be computed in the function class GapNC^1, and the n^{th} bit of every real root can be computed in TCLL. Finally, Lemma 4.13 ensures that we have computed the correct bit value. The argument for the binary case is analogous and uses Lemma 4.26 instead of lemmas 4.19, 4.23, and 4.24. $\square$

Proof of Theorem 4.1. Let α be a constant have series of the form $\alpha = \sum_{k=0}^{\infty} t_k = S_n + R_n$, where we have split the series into a finite sum $S_n = \sum_{k=0}^{n} t_k$ and a remainder series $R_n = \sum_{k=n+1}^{\infty} t_k$. Each term t_k of the series can be written as a rational number of the form $t_k = \beta^{-kc}\frac{p(k)}{q(k)}$, where β is a real number, $p(k), q(k)$ are fixed polynomials with integer coefficients and $c \geq 1$ is an integer. This series consist of a summation of iterated multiplication, division and addition which can be computed by TC^0 circuit. Since α has bounded irrationality measure, so its n^{th} bit can be computed using Lemma 4.13. $\square$

Using the BBP-like series for π [16] and its bounded irrationality measure, we get:

Corollary 4.33. *Computing the n^{th} bit of π is in TC^0 and $\mathsf{PH}^{\mathsf{PP}^{\mathsf{PP}^{\mathsf{PP}}}}$, given n in unary and binary respectively.*

4.5.1 Lower Bound

Finally, we show the Mod_p (for any odd prime p) hardness of the bits of a rational. We still don't have the proof of any kind of hardness of an irrational algebraic number.

Lemma 4.34. *For a given odd prime p and an integer X (having binary expansion $b_{n-1} \dots b_0$) there exist an integer N, whose bits are constructible by Dlogtime uniform projections, and a fixed rational number Q such that N^{th} digit in binary expansion of Q is 0 iff $\sum_i b_i \equiv 0 \pmod{p}$.*

Proof. For a given odd prime p we can find a integer t, $0 < t < p$, such that $2^t \equiv 1 \pmod{p}$. Such a t exists because the multiplicative group of integers modulo p is finite.

Consider the number $N = \sum_{i=0}^{n-1} b_i (2^t)^i$. Then, $N \equiv \sum_{i=0}^{n-1} b_i \pmod{p}$, because $2^t \equiv 1 \pmod{p}$. Now, consider the sum: $Q = \sum_{M>0} \frac{M \bmod p}{(2^t)^M}$. The N^{th} digit of Q is 0 iff $\sum_i b_i \equiv 0 \pmod{p}$.

For completeness sake, we compute,

$$\begin{aligned} Q &= \sum_{M \equiv 1 \bmod p} \frac{1}{(2^t)^M} + \sum_{M \equiv 2 \bmod p} \frac{2}{(2^t)^M} + \dots \sum_{M \equiv (p-1) \bmod p} \frac{(p-1)}{(2^t)^M} \\ &= \frac{(2^t)^p}{(2^t)^p - 1} \sum_{i=1}^{p-1} \frac{i}{(2^t)^i} \\ &= \frac{2^t(2^{tp} - 2^t p + p - 1)}{(2^t - 1)^2 (2^{tp} - 1)}. \end{aligned}$$

□

4.6 Conclusion and Open Problems

We take the first step in the complexity of Algebraic Numbers. Many questions remain open. We focus on fixed algebraic numbers - in general we could consider algebraic numbers defined by polynomials of varying degrees/coefficients. We have ignored complex algebraic numbers - they could present new challenges. Most importantly, our study is, at best, initial because of the enormous gap between lower bounds (virtually non-existent) and the upper bounds. Narrowing this gap is one of our future objectives.

Chapter 5

Summary

As we mentioned in the introduction, many data sets that arise in the fields of banking, telecommunications, biology, finance, climatology, artificial intelligence etc. are massive. In fact, they are so huge that even for reading the whole data, one may require very large resources. However, we live in a world of limited resources. At the same time, in order to meet our business needs, we may need to manage/analyse these massive data sets with the given limited resources. Managing and analyzing such data sets forces us to reconsider the traditional notions of efficient algorithms because processing such massive data sets even in linear time/space is far too expensive. Hence, there is a desire to develop algorithms which require *sublinear* resources. In this thesis, we have considered two major branches of sublinear algorithms: *sublinear time algorithms* and *sublinear space algorithms*. In sublinear time algorithms, we have developed efficient *property testing* algorithms for *promise* problems, while in sublinear space algorithms we have developed a space efficient algorithm in the Logspace model of computation.

In sublinear time algorithms, we have considered two different types of problems in property testing framework and have given sublinear time algorithms for their solutions. The first problem that we have considered belongs to the areas of computational geometry and cluster analysis (discussed in Chapter 2). The second problem belongs to the areas of distribution analysis and graph theory (discussed in Chapter 3). In sublinear space algorithms, we have considered a problem of computing bits of (real) algebraic number (discussed in Chapter 4).

In Chapter 2, we have considered one of the most fundamental problems of Computational Geometry, which is known as the clustering problem. Clustering is a common problem that arises in the analysis of large data sets. In a typical clustering problem, we have a set of n input points in a d dimensional space and the goal is to partition the points into k clusters, such that the radius of the maximum size cluster is minimized. This problem is computationally NP-hard.

We have considered this problem in the property testing framework. If S is a set of n points in $\mathbb{R}^d$, we say that S is (k, G)-clusterable if it can be partitioned into k clusters (subsets) such that each cluster is contained in a translated copy of a geometric object G. Here, given a set S of n points in $\mathbb{R}^d$ as input, we have presented a testing algorithm for (k, G)-clustering, ***i.e.***, an algorithm that can distinguish between the two cases: when S is (k, G)-clusterable, and when it is ϵ-far (where $0 < \epsilon \leq 1$) from being (k, G)-clusterable. A set S is ϵ-far from being (k, G)-clusterable if at least ϵn points need to be removed from S to make it (k, G)-clusterable.

For $k = 1$ and when G is a symmetric convex object, we have presented a testing algorithm which takes a constant sized sample from the input and tests the above property. Thus, we have proved that testing the above property is *testable* for this case. While the testing algorithm is simple, the proof of its correctness is a little involved for which we have used Helly's theorem.

For $k > 1$, we have solved a *weaker* version of this problem. The major bottleneck towards solving this problem exactly is that a Helly-type theorem is not known for the k object setting. We have stated a conjecture (Conjecture 2.12) that a similar type of Helly's theorem would also hold for the k object setting. Using a greedy algorithm, we have proved a weaker version of the conjecture (see Lemma 2.14). A major open problem of this thesis is to prove this conjecture.

Alon *et al.* [6] also considered a similar problem of testing (k, b)-*clustering**. For the $k = 1$ case, our result and the result from [6] give constant query testing algorithms. For the $k > 1$ case, we have given a *weaker* query complexity algorithm which works for $\epsilon \in (\epsilon', 1]$ where $\epsilon' = \epsilon'(k, t)$ (where t is a constant which depends on the shape of the geometric object). Alon *et al.* used the sophisticated VC-dimension technique while we have used Helly-type results. The main novelty of our result is that we have provided an interesting connection between Helly-type results and the clustering problem.

In Chapter 3, we have considered a problem which belongs to the areas of distribution analysis and graph theory. In particular, this problem is to test whether the stationary distribution obtained by a Markov chain on a directed graph is uniform. A random walk on a directed graph gives a Markov chain on the vertices of the graph. An important question that arises often in the context of a Markov chain is, whether the uniform distribution on the vertices of the graph is a stationary distribution of the Markov chain. Stationary distribution of a Markov chain is a global property of the graph. In this thesis, we have proved (contrary to the normal belief) that for a regular directed graph whether the uniform distribution on the vertices of the graph is a stationary distribution, depends on a "local property" of the graph, namely if (u, v) is a directed edge, then outdegree(u) is equal to indegree(v).

This result also has an application to the problem of testing whether a given distribution is uniform or "far" from being uniform. This is a well studied problem in the property testing and statistics. If the given distribution is the stationary distribution of the lazy random walk on a directed graph and the graph is given as an input, then how many bits of the input graph does one need to query in order to decide whether the distribution is uniform or "far" from it? This

*A set of points is said to be (k, b)-*clusterable* if it can be partitioned into k *clusters*, where radius (or diameter) of every cluster is at most b.

is a problem of graph property testing and we have considered this problem in the orientation model (introduced by Halevy *et al.* [45]). We have reduced this problem to testing (in the orientation model) whether a directed graph is Eulerian. Using a result of Fischer *et al.* [42] on the query complexity of testing Eulerianity (in the orientation model), we have obtained bounds on the query complexity for testing whether the stationary distribution is uniform. This result indicates that in the orientation model, a global property such as "the uniformity of stationary distribution" can be tested by testing a very local property of the graph.

This result holds only for graphs where the sum of in-degree and out-degree of all the vertices are same. Another major open problem of this thesis is to prove a similar statement for more general graphs.

In Chapter 4, we have initiated the complexity theoretic study of the problem of computing the bits of (real) algebraic numbers. Building on the surprising Bailey-Borwein-Plouffe (BBP) formula [17] for π, Yap [80] has shown that certain transcendental numbers such as π have binary expansions computable in a small complexity class like Deterministic Logspace. Motivated by his result, we have attempted to answer the corresponding question for algebraic numbers. Our main result is that computing a bit of a fixed real algebraic numbers is in GapNC^1 (a subclass of Logspace) when the bit position has a verbose (unary) representation, and in the counting hierarchy when the bit position has a succinct (binary) representation.

Our tools are drawn from elementary analysis and numerical analysis, and include the Newton-Raphson method. In order to keep our proofs simple, we have preferred to use the elementary Liouville's theorem over the much deeper Roth's theorem for algebraic numbers.

Jerábek [53] has also considered a similar problem of computing the n^{th} bit of an algebraic number. He has shown that for any constant d, the complex roots of a degree d univariate rational polynomial (given by a list of coefficients in binary) can be computed to a given accuracy by a uniform TC^0 circuit. Although for the verbose version of the problem his claimed upper bound is better than ours (as $\mathsf{TC}^0 \subseteq \mathsf{GapNC}^1$), his result does not pertain to the succinct version of the problem.

We have focused on fixed algebraic numbers only, proving similar results for algebraic numbers defined by polynomials of varying degrees/coefficients remains an open problem. We have ignored complex algebraic numbers - they could also present new challenges. We leave the possibility of proving non-trivial lower bounds for the problem of computing the bits of an algebraic number given the bit position in binary, as our main open question. In this direction, we have shown limited progress by proving a lower bound for *rationals*. Also, we have proved a weak upper bound for the succinct version of the problem, and improving this bound is another open problem.

Bibliography

[1] M. Abramowitz and I. A. Stegun. *Handbook of Mathematical Functions: with Formulas, Graphs and Mathematical Tables*. Dover, New York, 1972.

[2] Pankaj K. Agarwal and Cecilia Magdalena Procopiuc. Exact and approximation algorithms for clustering (extended abstract). *SODA*, pages 658–667, 1998.

[3] Pankaj K. Agarwal and Micha Sharir. Efficient algorithms for geometric optimization. *ACM Comput. Surv.*, 30(4):412–458, 1998.

[4] Eric Allender, Nikhil Balaji, and Samir Datta. Low-depth uniform threshold circuits and the bit-complexity of straight line programs. *Electronic Colloquium on Computational Complexity (ECCC)*, 20:177, 2013.

[5] Eric Allender, Peter Bürgisser, Johan Kjeldgaard Pedersen, and Peter Bro Miltersen. On the complexity of numerical analysis. *SIAM J. Comput.*, 38(5):1987–2006, 2009.

[6] Noga Alon, Seannie Dar, Michal Parnas, and Dana Ron. Testing of clustering. *SIAM J. Discrete Math.*, 16(3):393–417, 2003.

[7] Noga Alon and G. Kalai. A simple proof of the upper bound theorem. *European J. Combinatorics*, (6):211–214, 1985.

[8] Noga Alon and Michael Krivelevich. Testing k-colorability. *SIAM J. Discrete Math.*, 15(2):211–227, 2002.

[9] Noga Alon, Yossi Matias, and Mario Szegedy. The space complexity of approximating the frequency moments. *J. Comput. Syst. Sci.*, 58(1):137–147, 1999.

[10] M. R. Anderberg. *Cluster Analysis for Applications*. Academic Press, 1973.

[11] Sanjeev Arora and Boaz Barak. *Computational Complexity - A Modern Approach*. Cambridge University Press, 2009.

[12] Sanjeev Arora, Carsten Lund, Rajeev Motwani, Madhu Sudan, and Mario Szegedy. Proof verification and the hardness of approximation problems. *J. ACM*, 45(3):501–555, 1998.

[13] Sanjeev Arora and Shmuel Safra. Probabilistic checking of proofs: A new characterization of NP. *J. ACM*, 45(1):70–122, 1998.

[14] László Babai, Lance Fortnow, Leonid A. Levin, and Mario Szegedy. Checking computations in polylogarithmic time. In *STOC*, pages 21–31, 1991.

[15] László Babai, Lance Fortnow, and Carsten Lund. Non-deterministic exponential time has two-prover interactive protocols. *Computational Complexity*, 1:3–40, 1991.

[16] David H. Bailey. A compendium of BBP-type formulas for mathematical constants. Report, Lawrence Berkeley National Laboratory, Berkeley, CA, USA, February 2011.

[17] David H. Bailey, Jonathan M. Borwein, Peter B. Borwein, and Simon Plouffe. The quest for pi. *The Mathematical Intelligencer*, 19(1):50–57, January 1997.

[18] David A. Mix Barrington, Peter Kadau, Klaus-Jörn Lange, and Pierre McKenzie. On the complexity of some problems on groups input as multiplication tables. *J. Comput. Syst. Sci.*, 63(2):186–200, 2001.

[19] Tugkan Batu, Sanjoy Dasgupta, Ravi Kumar, and Ronitt Rubinfeld. The complexity of approximating the entropy. *SIAM J. Comput.*, 35(1):132–150, 2005.

[20] Tugkan Batu, Lance Fortnow, Eldar Fischer, Ravi Kumar, Ronitt Rubinfeld, and Patrick White. Testing random variables for independence and identity. In *FOCS*, pages 442–451, 2001.

[21] Tugkan Batu, Lance Fortnow, Ronitt Rubinfeld, Warren D. Smith, and Patrick White. Testing that distributions are close. In *FOCS*, pages 259–269, 2000.

[22] Tugkan Batu, Lance Fortnow, Ronitt Rubinfeld, Warren D. Smith, and Patrick White. Testing closeness of discrete distributions. *J. ACM*, 60(1):4, 2013.

[23] Mihir Bellare, Don Coppersmith, Johan Håstad, Marcos A. Kiwi, and Madhu Sudan. Linearity testing in characteristic two. *IEEE Transactions on Information Theory*, 42(6):1781–1795, 1996.

[24] Mihir Bellare, Oded Goldreich, and Madhu Sudan. Free bits, PCPs, and non-approximability-Towards tight results. *SIAM J. Comput.*, 27(3):804–915, 1998.

[25] Mihir Bellare, Shafi Goldwasser, Carsten Lund, and A. Russeli. Efficient probabilistically checkable proofs and applications to approximations. In *STOC*, pages 294–304, 1993.

[26] Mihir Bellare and Madhu Sudan. Improved non-approximability results. In *STOC*, pages 184–193, 1994.

[27] Michael Ben-Or, Ephraim Feig, Dexter Kozen, and Prasoon Tiwari. A fast parallel algorithm for determining all roots of a polynomial with real roots. *SIAM J. Comput.*, 17(6):1081–1092, 1988.

[28] Manuel Blum, Michael Luby, and Ronitt Rubinfeld. Self-testing/correcting with applications to numerical problems. *J. Comput. Syst. Sci.*, 47(3):549–595, 1993.

[29] Sourav Chakraborty, Eldar Fischer, Oded Lachish, Arie Matsliah, and Ilan Newman. Testing *st* -connectivity. In *APPROX-RANDOM*, pages 380–394, 2007.

[30] Sourav Chakraborty, Akshay Kamath, and Rameshwar Pratap. Testing uniformity of stationary distribution. *CoRR*, abs/1302.5366, 2013.

[31] Sourav Chakraborty, Akshay Kamath, and Rameshwar Pratap. Testing uniformity of stationary distribution. In *Eurocomb*, pages 589–594, 2013.

[32] Sourav Chakraborty, Rameshwar Pratap, Sasanka Roy, and Shubhangi Saraf. Helly-type theorems in property testing. In *LATIN*, pages 306–317, 2014.

[33] Moses Charikar, Samir Khuller, David M. Mount, and Giri Narasimhan. Algorithms for facility location problems with outliers. pages 642–651, 2001.

[34] F. R. K. Chung. *Spectral Graph Theory*. American Mathematical Society, 1997.

[35] Artur Czumaj and Christian Sohler. Property testing with geometric queries. In *ESA*, pages 266–277, 2001.

[36] Artur Czumaj, Christian Sohler, and Martin Ziegler. Property testing in computational geometry. In *ESA*, pages 155–166, 2000.

[37] L. Danzer and B. Grünbaum Branko. Intersection properties of boxes in $\mathbb{R}^d$. *Combinatorica*, 2(3):237–246, 1982.

[38] Samir Datta and Rameshwar Pratap. Computing bits of algebraic numbers. In *TAMC*, pages 189–201, 2012.

[39] J. Eckhoff. An upper bound theorem for families of convex sets. *Geom. Dediata 19*, (75):217–227, 1985.

[40] Uriel Feige, Shafi Goldwasser, László Lovász, Shmuel Safra, and Mario Szegedy. Approximating clique is almost NP-complete (preliminary version). In *FOCS*, pages 2–12, 1991.

[41] Eldar Fischer. The art of uninformed decisions: A primer to property testing. *Current Trends in Theoretical Computer Science: The Challenge of the New Century, G. Paun, G. Rozenberg and A. Salomaa (editors), World Scientific Publishing*, I:229–264, 2004.

[42] Eldar Fischer, Oded Lachish, Ilan Newman, Arie Matsliah, and Orly Yahalom. On the query complexity of testing orientations for being Eulerian. In *APPROX-RANDOM*, pages 402–415, 2008.

[43] Oded Goldreich. Combinatorial property testing (a survey). *Electronic Colloquium on Computational Complexity (ECCC)*, 4(56), 1997.

[44] Oded Goldreich, Shafi Goldwasser, and Dana Ron. Property Testing and its Connection to Learning and Approximation. *J. ACM*, 45(4):653–750, 1998.

[45] Shirley Halevy, Oded Lachish, Ilan Newman, and Dekel Tsur. Testing properties of constraint-graphs. In *IEEE Conference on Computational Complexity*, pages 264–277, 2007.

[46] G. H. Hardy and E. M. Wright. *An Introduction to the Theory of Numbers*. Oxford Univ. Press, New York, 5th edition, 1979.

[47] Johan Håstad. Testing of the Long Code and Hardness for Clique. In *STOC*, pages 11–19, 1996.

[48] Johan Håstad. Some optimal inapproximability results. In *STOC*, pages 1–10, 1997.

[49] E. Helly. Über Mengen konvexer Köper mit gemeinschaftlichen Punkten (germen). *Jahresber. Deutsch.Math. Verein.*, (32):175–176, 1923.

[50] William Hesse, Eric Allender, and David A. Mix Barrington. Uniform constant-depth threshold circuits for division and iterated multiplication. *J. Comput. Syst. Sci.*, 65(4):695–716, 2002.

[51] A. Hume. *Handbook of massive data sets*, chapter Billing in the large, pages 895–909. Kluwer Academic Publishers, 2002.

[52] A. K. Jain and R. C. Dubes. *Algorithms for Clustering*. Prentice-Hall, 1988.

[53] Emil Jerábek. Root finding with threshold circuits. *Theor. Comput. Sci.*, 462:59–69, 2012.

[54] G. Kalai. Intersection patterns of convex sets. *Israel J. Math.*, (48):161–174, 1984.

[55] M. Katchalski and D. Nashtir. On a conjecture of Danzer and Grunbaum. *Proc. A.M.S*, (124):3213–3218, 1996.

[56] L. Kaufman and P. J. Rousseeuw. *Finding Groups in Data: An Introduction to Cluster Analysis*. John Wiley, 1990.

[57] Harry R. Lewis and Christos H. Papadimitriou. Symmetric space-bounded computation. *Theor. Comput. Sci.*, 19:161–187, 1982.

[58] L. Lovász. Random walks on graphs: A survey. *Combinatorics, Paul Erdos is Eighty*, 2(1):1–46, 1993.

[59] Alexis Maciel and Denis Thérien. Threshold circuits of small majority-depth. *Inf. Comput.*, 146(1):55–83, 1998.

[60] A. Markoff. Sur une question posée par Mendeleieff. *Bulletin of the Academy of Sciences of St. Petersburg*, 62:1–24, 1889.

[61] Nimrod Megiddo. The weighted euclidean 1-center problem. *Mathematics of Operations Research - MOR.*, 8(4):498–504, 1983.

[62] M.Katchalski and A. Liu. A problem of geometry in $\mathbb{R}^n$. *Proc. A.M.S*, (75):284–288, 1979.

[63] R. W. Moore. Enabling petabyte computing. `http://www.nap.edu/html/whitepapers/ch-48.html`, 2000.

[64] S. Muthukrishnan. Data streams: Algorithms and applications. *Foundations and Trends in Theoretical Computer Science*, 1(2), 2005.

[65] C. Andrew Neff and John H. Reif. An efficient algorithm for the complex roots problem. *J. Complexity*, 12(2):81–115, 1996.

[66] Oystein Ore. On functions with bounded derivatives. *Transactions of the American Mathematical Society*, 43(2):pp. 321–326, 1938.

[67] Omer Reingold. Undirected connectivity in Logspace. *J. ACM*, 55(4), 2008.

[68] V. Rödl and R. Duke. On graphs with small subgraphs of large chromatic number. *Graphs and Combinatorics*, 1:91–96, 1985.

[69] Dana Ron. Property testing (foundations and trends in machine learning). *Publishers: Now Publishers Inc.*

[70] Dana Ron. Property testing: A learning theory perspective. *Foundations and Trends in Machine Learning*, 1(3):307–402, 2008.

[71] Klaus Friedrich Roth. Rational approximations to algebraic numbers. *Mathematika. A Journal of Pure and Applied Mathematics*, 2:1–20, 1955.

[72] Ronitt Rubinfeld. On the robustness of functional equations. *SIAM J. Comput.*, 28(6):1972–1997, 1999.

[73] Ronitt Rubinfeld and Madhu Sudan. Robust characterizations of polynomials with applications to program testing. *SIAM J. Comput.*, 25(2):252–271, 1996.

[74] I. Z. Ruzsa and Endre Szemerédi. Triple systems with no six points carrying three triangles. *Combinatorics (Keszthely), Coll. Math. Soc. J. Bolyai*, 18(2):939–945, 1976.

[75] A. B Shidlovskii. *Transcendental Numbers*. de Gruyter, New York, 1989.

[76] Luca Trevisan. Recycling queries in PCPs and in linearity tests (extended abstract). In *STOC*, pages 299–308, 1998.

[77] A.M. Turing. On computable numbers, with an application to the Entscheidungs problem. *Proc. London Math. Soc.*, 2(42):230–265, 1936.

[78] Heribert Vollmer. *Introduction to circuit complexity - a uniform approach*. Texts in theoretical computer science. Springer, 1999.

[79] Chee Yap. *Fundamental Problems in Algorithmic Algebra*. Oxford University Press, 2000.

[80] Chee Yap. Pi is in Logspace. Manuscript, June 2010.