Non-monotonicity, Projective Dimension and Circuit Depth and Size Lower Bounds

A THESIS

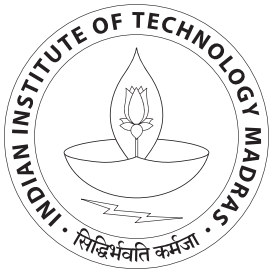
submitted by

SAJIN KOROTH

for the award of the degree

of

DOCTOR OF PHILOSOPHY



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY, MADRAS.

July 2017

THESIS CERTIFICATE

This is to certify that the thesis entitled **Non-monotonicity, Projective Dimension and Circuit Depth and Size Lower Bounds**, submitted by **Sajin Koroth**, to the Indian Institute of Technology, Madras, for the award of the degree of **Doctor of Philosophy**, is a bona fide record of the research work carried out by him under my supervision. The contents of this thesis, in full or in parts, have not been submitted to any other Institute or University for the award of any degree or diploma.

Dr. Jayalal Sarma
Research Guide
Associate Professor
Dept. of Computer Science and Engineering
IIT-Madras, 600 036

Place: Chennai

Date: July 10, 2017

ACKNOWLEDGEMENTS

I thank my advisor Jayalal Sarma who was an integral part of this journey. I am very thankful for the great discussions and the long hours he spent with me during the initial years. Later in my PhD when I had to face a serious personal crisis he was an immense source of support which helped me continue on. I really admire his work ethic and his immense attention to detail. He has been instrumental in fighting the bureaucracy and making sure that everything went smoothly. Though we have our philosophical differences, he was instrumental in making me introspect the many subtle aspects of research life and work ethic. I also appreciate the immense patience he had for providing me a space to grow and ensuring my growth despite my many shortcomings.

I am also thankful for the many discussions I had with people who I had the opportunity of working with during my PhD. I am very thankful to my collaborators who contributed directly to my thesis, Krishnamoorthy Dinesh and Jayalal Sarma and many wonderful discussions we had. A big thanks to Krishnamoorthy Dinesh with whom I had many discussions about research and life. I really enjoyed his company and his infinite enthusiasm towards theoretical computer science. I am extremely thankful to Rocco Servedio for discussions which helped shape one of the chapters in this thesis. I am thankful to Swastik Kopparty, Shubhangi Saraf and Alexander Kulikov for hosting at various times and many great discussions and continued support. I am really thankful to Or Meir for mentoring me during a long visit and helping me grow a lot as researcher. I also thank my thesis committee for

their valuable feedback and support.

I thank N.S. Narayanaswamy and Raghavendra Rao for their support in many aspects of my PhD life. I am also very thankful for my masters thesis advisor Shankar Balachandran for his continued support as a mentor. I am thankful for the courses which made me passionate about theoretical computer science, both at IITM and at the Institute of Mathematics Sciences (IMSc). I would like to thank V. Arvind, Saket Saurabh, Meena Mahajan, Prahladh Harsha, R. Ramanujam, C. R. Subramanian, N.S. Narayanaswamy, Shankar Balachandran, Jayalal Sarma and Raghavendra Rao for the wonderful courses they offered. I would especially like to thank Saket Saurabh for his mentoring.

I am also thankful for the student community in Chennai for providing a warm and enthusiastic academic environment. I would like to thank especially, Sadagopan Narasimhan, Krishnamoorthy Dinesh, R. Subhashini, G. Ramakrishna, R. Krithika, Nikhil Balaji, Nitin Saurabh, Raja S, Syed Meesum, Ramantahn Thinniyam and Suryajith Chillara. I am thankful to many of my officemates who were a source of great friendship, good working environment and great many coffee meet-ups. I would like to thank, Krishnamoorthy Dinesh, Ramya Chandrasekhar, William Moses, Sumathi Sivasubrahmaniam, Purnata Ghoshal and Ankit Chauhan.

I would like to thank my friends for their great support. I am really thankful to my best friend Murali Krishna Karnam who has always been there for me. I would also like to thank many friends who offered great support and many memorable conversations, especially, Anup Pydah, William Moses, Saurav Chandra and Nitin Sanghe.

Lastly I would like to thank my family. I would like to thank my father P. P.

Sreedharan who offered such unconditional love and warmth, I have no words to describe them. I deeply mourn his loss which happened during my undergraduate days. I suffered great personal loss during my PhD, I almost lost my entire world. I lost my dear grandpa P. Balan and my dear mother P. K. Gomathy. My grandpa was an inspiration for me while growing up. We were like two inseparable friends. He was my spiritual guru and my greatest companion. My mother was my entire world. She was a strong independent woman who struggled through her entire life, lived an ideal and principled life and loved me dearly. There was almost nothing I did not share with her. She pushed through so many difficulties to see me fulfill my dreams, to the point of not telling me about an abnormal growth in the worry of affecting my studies, which finally turned out to be a terminal cancer which consumed her. I would not have survived these events if not for my soul-mate Esha Ghosh. She brought meaning back to a world which had lost all meaning. She did not just bring me back to life, she enriched my soul by such love and introspection, I have no words to describe the beautiful spiritual growth I had with her and continuing to have. It has been a wonderful journey together since our first meeting as prospective masters students. She also worked miracle by almost bringing back my lost family, by bringing in her family. I am also thankful to my mother Sarbari Ghosh, my father Somanath Ghosh and my grandma Tripti Das. This journey would have been impossible without them. I would also like to thank Chaya Ganesh who has been more than friend, spiritual companion and has become family for her support and crucial role in my life. I would also like to thank Salini aunty and Pradeepettan for helping, loving and taking care of my family.

DEDICATION

to my mother Gomathy, who was my whole world

ABSTRACT

KEYWORDS: Complexity Theory, Non-monotone Circuits, Depth Lowerbounds, Learning Non-monotone Circuits, Branching Programs, Projective Dimension, Size Lowerbounds

Two central resources of computation in complexity theory are time and space. We study central problems about these resources using combinatorial models of computation. We first study Boolean circuits a combinatorial model of computation which captures parallel time needed for computation. And then we study branching programs, another combinatorial model of computation, which captures space needed for computation.

We first study depth lower bounds against non-monotone circuits, parametrized by a new measure of non-monotonicity: the orientation of a function f is the characteristic vector of the minimum sized set of negated variables needed in any DeMorgan circuit (circuits where negations appear only at the leaves) computing f. We prove trade-off results between the depth and the weight/structure of the orientation vectors in any circuit C computing the **CLIQUE** function on an n vertex graph. We prove that if C is of depth d and each gate computes a Boolean function with orientation of weight at most w (in terms of the inputs to C), then $d \times w$ must be $\Omega(n)$. In particular, if the weights are $o(\frac{n}{\log^k n})$, then C must be of depth $\omega(\log^k n)$. We prove a barrier for our general technique. However, using specific properties of the **CLIQUE** function (used in Amano Maruoka (2005)) and the Karchmer–Wigderson

framework (Karchmer Wigderson (1988)), we go beyond the limitations and obtain lower bounds when the weight restrictions are less stringent. We then study the depth lower bounds when the structure of the orientation vector is restricted. Asymptotic improvements to our results (in the restricted setting) separates NP from NC. As our main tool, we generalize Karchmer–Wigderson games (Karchmer Wigderson (1988)) for monotone functions to work for non-monotone circuits parametrized by the weight/structure of the orientation. We also prove structural results about orientation and prove connections between number of negations and weight of orientations required to compute a function.

Using characterization of minimal orientation of a function, and the learning algorithm of [BCO+15a] we come up with a learning algorithm under the uniform distribution membership query model for learning n variable functions computed by Boolean circuits whose weight of orientation is at most w to error ϵ in time $n^{O(w\sqrt{n}/\epsilon)}$. [BCO+15a] prove a near matching learning lower bound for a family functions $\mathcal{H}^k$ of balanced $k(n)$ alternating functions Using characterization of orientation, we show that this family of functions $\mathcal{H}^k$ has minimal weight of orientation equal to the number of inputs. Based on the techniques employed in the lower bound of [BCO+15a], the hardness amplification of learning composition of functions based on a Fourier analytic property Expected Bias ([FLS11], [O'D04]), we come up with information theoretic noise model tailored for sparse orientation. Based on the intuition behind the noise model we prove a theorem which supports expected bias as the right parameter for studying hardness amplification under composition of certain class of functions. We also state a conjecture based on this intuition for hardness amplification under composition of functions based on the noise model. We also state open problems whose solution along with the conjecture would yield lower bounds for learning sparsely oriented circuits.

We then study branching program lower bounds using projective dimension, a graph parameter (denoted by $\mathsf{pd}(G)$ for a graph G), introduced by Pudlák and Rödl (1992). For a Boolean function f(on n bits), Pudlák and Rödl associated a bipartite graph G_f and showed that size of the optimal branching program computing f (denoted by $\mathsf{bpsize}(f)$) is at least $\mathsf{pd}(G_f)$ (also denoted by $\mathsf{pd}(f)$). Hence, proving lower bounds for $\mathsf{pd}(f)$ imply lower bounds for $\mathsf{bpsize}(f)$. Despite several attempts (Pudlák and Rödl (1992), Rónyai et.al, (2000)), proving super-linear lower bounds for projective dimension of explicit families of graphs has remained elusive. We observe that there exist a Boolean function f for which the gap between the $\mathsf{pd}(f)$ and $\mathsf{bpsize}(f)$) is $2^{\Omega(n)}$. Motivated by the argument in Pudlák and Rödl (1992), we define two variants of projective dimension - *projective dimension with intersection dimension 1* (denoted by $\mathsf{upd}(f)$) and *bitwise decomposable projective dimension* (denoted by $\mathsf{bpdim}(f)$). We show the following results :

(a) We observe that there exist a Boolean function f for which the gap between $\mathsf{upd}(f)$ and $\mathsf{bpsize}(f)$ is $2^{\Omega(n)}$. In contrast, we also show that the bitwise decomposable projective dimension characterizes size of the branching program up to a polynomial factor. That is, there exists an $0 < \epsilon < 1$ such that for any function f,

$$\mathsf{bpdim}(f)/6 \leq \mathsf{bpsize}(f) \leq (\mathsf{bpdim}(f))^{3+\epsilon}$$

(b) We introduce a new candidate function family f for showing super-polynomial lower bounds for $\mathsf{bpdim}(f)$. As our main result, we demonstrate gaps between $\mathsf{pd}(f)$ and the above two new measures for f :

$$\mathsf{pd}(f) = O(\sqrt{n}) \qquad \mathsf{upd}(f) = \Omega(n) \qquad \mathsf{bpdim}(f) = \Omega\left(\frac{n^{1.5}}{\log n}\right)$$

(c) Although not related to branching program lower bounds, we derive exponential lower bounds for two restricted variants of $\mathsf{pd}(f)$ and $\mathsf{upd}(f)$ respectively by observing that they are equal to well-studied graph parameters - bipartite clique cover number and bipartite partition number respectively.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABBREVIATIONS

pd	Projective dimension
upd	Projective dimension with intersection dimension 1
bitpdim	Bitwise decomposable projective dimension
NC	Nick's Class
BP	Branching Program

CHAPTER 1

Introduction

Complexity theory is a branch of theoretical computer science which studies inherent limits of computation by placing bounds on important resources of computation like time, space etc. Such a theoretical study is built on the foundation of a robust mathematical model of computation, which is simple yet elegant, of Turing [Tur36] called "Turing machines". This formalization of computing is so fundamental that it not only models, what we think is computable, but also models what is efficient to compute in an asymptotic sense. The abstraction allows one to ignore the specifics of the implementation of real computer like speed of the processor or the storage size. But it comes with the caveat that the abstraction captures the notion of efficiency only asymptotically. That is, it only captures the asymptotic growth of resource requirements as a function of input length. But this problem turns out to be more of a boon than a curse as this makes the definition of efficiency robust and independent of the specifics of the implementation of an actual computing device. And this notion of asymptotic growth is required as we do not want to build a theory of computation which classifies problems as efficient or not based on the current technology. As it is evident from history, computation power and resources has been growing at an almost exponential rate (see Moore's Law [BM06]). Also, most of the algorithms which are efficient asymptotically are also efficient in practice. But more importantly, for the natural problems we would like to solve but have been unable to, we do not know efficient algorithms even in the asymptotic sense.

Two major resources of computation which are interesting from both practical and theoretical view points, are time and space. In this thesis, we study central problems in complexity theory related to both of these resources.

We will first talk about time as a resource. A robust notion of efficient time for computation, as suggested by Edmonds [Edm65], is the class P. Class P is a set of problems which can be solved on a computer (more formally a Turing machine) within fixed polynomial time of the input length. This class encompasses many natural problems like sorting a given set of n numbers. An interesting class of problems which are believed to be not all of P is the class of problems which have efficient parallel solutions. This class and its relation to P is naturally an important question in complexity theory. And it is even more interesting for the fact that this problem can be stated as a problem about a combinatorial model of computation, called Boolean circuits. Such a model of computation is deemed helpful in settling questions about computation as it brings about tools from combinatorics.

An interesting class of problems which are not known to be in P, but nonetheless is very important in terms of both theory and practice is the class NP. An example of a problem in class NP is the problem **CLIQUE**(n, k), which given an undirected graph on n vertices, asks whether there is a k-clique, i.e., a set of k vertices in which every pair of vertices is connected by an edge. It is not known how to search for existence a k-clique in a given graph in polynomial time for large values of k like $k = n/2$. But given an input instance, i.e., an undirected graph G, and a set of k vertices of G as a candidate certificate for a k-clique in G, it can be verified in polynomial time whether these vertices form a k-clique or not.

Class NP abstracts languages which are hard to solve, but for which for any input, given a certificate of membership of polynomial size in the input, it can be

verified in polynomial time whether it is valid certificate of membership. And also for inputs which are in the language there is always a certificate of polynomial size certifying the membership of the input. The general definition of class NP is thus the set of all problems which have an efficient certificate (i.e., a fixed polynomial in input size) and given such a certificate its validity can be verified in polynomial time. This class of problems is modeled using the notion of non-determinism in computation. More formally, NP corresponds to the class of problems solvable in non-deterministic polynomial time on Turing machines, as a polynomial size certificate can be guessed using non-determinism in non-deterministic polynomial time, and then it can be verified whether the non-deterministically guessed certificate is valid or not in deterministic polynomial time. A central problem in complexity theory is to understand the power of non-determinism and its relation to efficient deterministic computation. It is widely believed that class NP different from class P, i.e., there are problems which can be solved in polynomial time using non-determinism but which cannot be solved in deterministic polynomial time. Very little progress has been made towards settling this central problem. It is not even known if hard problems in NP do not have efficient parallel algorithms. But there is good reason to believe that NP is different from P. In fact, we know that if NP is indeed equal to P, many complexity classes which are believed to be separate would collapse on to one another. Scott Aaronson has recently published a detailed survey ([Aar17]) of the P vs NP problem which includes recent progress on the problem and also why P is believed to be different from NP.

1.1 Circuit depth and the P vs NC problem

In the early 90's a lot of progress was made towards separation of NP from efficient parallel time by studying a combinatorial model of computation called Boolean circuits. This sub-area of complexity theory is called circuit complexity. And it models computation as Boolean circuits, a non-uniform model of computation. A Boolean circuit is a directed acyclic graph (DAG) with a designated node called the root node. The in-degree of each node in the circuit is called fan-in and out-degree of each node in the circuit is called fan-out. The root node has fan-out 0. A node which has in-degree 0 is called a "leaf". Every node which is not a leaf node is called an internal node. A node u is said to be the child of a node v if there is a directed edge from u to v. The leaf nodes of the circuit are labeled by input variables $\{x_i\}_{i\in[n]}$. Every internal node is labeled by a Boolean function $f : \{0,1\}^r \to \{0,1\}$ from a set of allowed functions called basis of the circuit, where r is the fan-in of the node. The function computed by a circuit is recursively defined. The function computed by a leaf node is the variable x_i where x_i is the label of the node. The function computed by an internal node is the function f applied to the functions computed by its child nodes, where f is the function labeling the current node. The Boolean function computed by the circuit is the function computed by the root node. One of the central complexity resources associated with a circuit is its size, which is the number of nodes in the given circuit. It is a non-uniform model of computation as a specific circuit is defined over a certain number of inputs, and hence by definition of the model there are different circuits for different input lengths. A Boolean function f is said to be computed by a family of circuits $C = \{C_n\}_{n>n_0, n\in\mathbb{N}}$ if there exists an $n_0 \in \mathbb{N}$ such that for any $n > n_0$ the nth slice of f, $f_n : \{0,1\}^n \to \{0,1\}$ is computed by C_n. This is unlike the description of a Turing machine which has a single uniform

description for all input lengths. It is also not necessary that there is an algorithm (or more formally a single Turing machine) producing the description of the circuit given the input length. This non-uniformity adds certain strange powers to the circuit like the ability to compute all sparse languages, including the unary halting problem which is not even computable by Turing machines.

But despite this power of non-uniformity, unless certain complexity classes which are believed to be separate collapse to one another, it is known ([KL82]) that NP-hard problems like **CLIQUE**$(n, n/2)$ cannot be computed by a family of poly-sized circuits. Poly-size circuits are represented by the class P/poly. If it can be shown that NP is separate from P/poly then it would also separate NP from P. This is because every polynomial time Turing machine can be converted to a polynomial sized circuit, implying that P $\subseteq$ P/poly. Circuit complexity is deemed hopeful for proving such lower bounds, as it is a combinatorial model of computation and allows us to use a vast array of techniques and tools from combinatorics. This hope was justified by early circuit lowerbounds, the like of [Raz85a] which used the Sunflower lemma ([ER60]) from combinatorics to prove circuit lower bounds.

A central resource of a Boolean circuit is its depth. Formally, depth is the length of the longest path from the output of a circuit to any of its inputs. Intuitively depth captures "parallel time" in Turing machines. Formally, it is known that depth of a circuit corresponds to time taken by a model of parallel computation called CREW PRAM ([JÓ92]). Intuitively, this is because if every node is considered as a processor and abstracts the computation time taken at a node to be a unit, then the longest time a node has to wait till its children produce inputs is at most the depth of the node.

The notion of "efficient parallel time" in circuits where internal nodes have bounded fan-in is captured by poly-logarithmic depth and polynomial size. This class of circuits are denoted by NC. Since $\mathsf{NC} \subseteq \mathsf{P/poly}$ it is believed that $\mathsf{NP} \not\subseteq \mathsf{NC}$. It is also believed that there are problems in P which do not have "efficient parallel algorithms". Thus, it is even believed that $\mathsf{P} \not\subseteq \mathsf{NC}$. There are "complete" (i.e., problems which are in P and such that every other problem in P can be reduced to an instance of this problem) problems for P, and they are not believed to be in NC. There is an interesting subclass of NC which models "truly efficient" parallel algorithms. This is the class NC^1 of polynomial sized, bounded fan-in and logarithmic depth circuits. There are problems which are not believed to be P complete, like Perfect Matching, which are in NC but are believed not to be in NC^1. The perfect matching problem asks whether given an undirected graph is there a set of edges such that any vertex in the graph appears as an end point of exactly one edge in the set. Though such problems are believed to separate NC from NC^1, we do not yet know of any separation between, even NP and NC^1.

1.1.1 Limited negation circuits

In the early 90's a lot of progress was made towards proving lower bounds against a restricted class of circuits called *monotone* circuits ([Raz85a, Raz85b, KW88, RW92]). A circuit is said to be monotone if all its internal gates are either AND gates or OR gates, and they do not have NOT gates. These circuits compute Boolean functions which are monotone, i.e., functions whose value does not decrease when input bits are changed from 0 to 1. If the undirected adjacency matrix of a graph is given as the input, changing an input bit from 0 to 1 is equivalent to adding the corresponding edge in the graph. Thus, the **CLIQUE**(n, k) function, which is NP-

complete for large k, is a monotone function. This is because adding edges cannot remove an existing clique. Monotone circuits do not allow NOT gates and can intuitively be thought of as representing computation without cancellations. An equivalent definition of a monotone circuit is that it is a circuit where every sub-circuit rooted at a gate computes a monotone function. Razborov [Raz85a] in his seminal lower bound proved that monotone circuits computing the **CLIQUE**(n, k) function requires size $\Omega(n^k)$ for any $k \leq \log n$. Later Alon and Boppana ([AB87]) extended it to a truly exponential size lower bound for the clique function by proving that detecting cliques of size $(1/4)(\frac{m}{\log m})^{2/3}$ in an m-vertex graph requires monotone circuits of size $\exp(\Omega(m/\log m)^{1/3})$.

One natural way to extend these lower bounds for monotone circuits to general circuits is to study circuits where the non-monotonicity is limited. Since monotone circuits are circuits without negation gates, a natural measure which parametrizes the amount of non-monotonicity in the circuit is the number of negation gates allowed in the circuit. Markov ([Mar58]) established that it is indeed a robust measure of non-monotonicity by proving that the minimum number of negations needed to compute a Boolean function is the property of the function under consideration. Markov ([Mar58]) proved that for a given Boolean function $f : \{0,1\}^n \to \{0,1\}$, the minimum number of negations needed to compute the function is independent of any circuit computing the function. He called this number the inversion complexity of a Boolean function and proved that for bounded fan-in circuits, it only depends on the function's non-monotone behavior on the hypercube, and is independent of the actual circuit computing it. In particular he proved that the inversion complexity of f denoted by $I(f)$ is well defined and is equal to $\lceil(\log(a(f) + b))\rceil - 1$ where $b = 1$ if $f(0^n) = 0$ and $b = 2$ otherwise [Mar58]. The measure $a(f)$ is a property of the function and is equal to the maximum number

of times the functions value changes from 1 to 0 on any chain of the hypercube. In 2005, Amano and Maruoka [AM05] showed how to extend the super polynomial size lower bound of Razborov to circuits with limited non-monotonicity. They parametrized non-monotonicity by the maximum number of NOT gates allowed in the circuit. They proved that any circuit computing **CLIQUE**$(n, n/2)$ with at most $\frac{1}{6}\log\log n$ negations requires super polynomial size. They used a combinatorial argument called monotone covering and combined it with Razborov's size lower bound. From a well known result of Fischer ([Fis75]) it is known that if a function is computed by a poly-size, log-depth circuit then it is also computed a circuit of poly-size, log-depth and at most $\log n$ negations. Hence to separate NP from P/poly it is enough to extend the lower bound of Amano Maruoka to $\log n$ negations.

Razborov also proved a super polynomial size lower bound [Raz85b] against monotone circuits computing perfect matching problem (denoted by **PMATCH**). Razborov's lower bound implied that no polynomial sized, poly-log depth monotone circuit can compute **PMATCH**. But it was left open whether super-polynomial sized monotone circuits of poly-log depth can compute **PMATCH**. Eva Tardos ([Tar88]) established a truly exponential size lower bound against monotone circuits computing a function which is in P. Thus, she settled the question by establishing a truly exponential gap between monotone circuit size and general circuit size. But the such a separation for monotone depth and general circuit depth was not known. This problem was later settled in the negative by a series of works by Karchmer, Raz and Wigderson ([KW88, RW92]) who proved that monotone circuits computing **CLIQUE**$(n, n/2)$ and **PMATCH** requires linear depth. This is an interesting result as this gives hopes of proving limited non-monotonicity depth lower bounds against even functions in P for which we do not know exponential mono-

tone circuit size lower bounds. Note that to get a super poly-logarithmic depth lower bound for a function using the framework of Amano Maruoka [AM05], one needs to prove exponential size lower bounds against monotone circuits computing that function. But for a function like **PMATCH** such a lower bound is not known. Hence a depth lower bound which does not depend on size lower bounds can potentially give depth lower bounds against "limited non-monotone circuits".

Raz and Wigderson ([RW92]) proved the linear depth lower bounds on **CLIQUE**$(n, n/2)$ and **PMATCH** using a bridge between circuit complexity and communication complexity called Karchmer Wigderson relations, introduced by Karchmer and Wigderson ([KW88]). The Karchmer Wigderson relation related to a Boolean function $f : \{0,1\}^n \to \{0,1\}$ captures the minimal depth of any circuit computing f. In this relation there are two players Alice and Bob who are given inputs $x \in f^{-1}(0)$ and $y \in f^{-1}(1)$ respectively, and the objective of the players is to output an index $i \in [n]$ such that $x_i \neq y_i$. The communication complexity of the best deterministic protocol solving this relation is equal to the circuit depth of the function f. The communication complexity of a deterministic protocol is the maximum number of bits communicated by the protocol over any input to the protocol. For monotone functions f, they also defined a relation which captures the monotone circuit depth of f. It is communication complexity of the best deterministic protocol where players Alice and Bob are given inputs $x \in f^{-1}(0)$ and $y \in f^{-1}(1)$ respectively, and the goal is to output an index $i \in [n]$ such that $x_i = 0, y_i = 1$. The linear lower bounds for depth of **CLIQUE**$(n, n/2)$, **PMATCH** is obtained by reducing the famous DISJ_n problem to the monotone Karchmer Wigderson relations corresponding to **CLIQUE**$(n, n/2)$ and **PMATCH**.

But despite this initial enthusiasm and some great results on restricted class of

Boolean circuits the progress towards original general questions has been almost stagnant. Complexity theorists have been able to explain the lack of progress by showing the impossibility of separating P and NP using current tools and techniques. More specifically Razborov and Rudich ([RR97]) showed that even separating NP and NC is beyond the reach of current techniques. They called the limitation of these techniques the "Natural proofs" barrier and defined what does it mean for a circuit lower bound proof to be "natural". They [RR97] showed that the celebrated exponential size lower bounds against monotone circuits and small depth circuits like the exponential size lower bound for constant depth unbounded fan-in circuits computing the parity function ([Smo87, Raz87]) over $\wedge, \vee, \neg$ basis are all "natural" proofs. They showed that unless sub-exponential strong one-way functions do not exist, these natural proofs cannot prove super polynomial lower bounds against many classes of circuits like NC^1 which are believed to contain such one-way functions. A one-way function is a function which can be evaluated in polynomial time given an input, but it is such that it is hard to invert. That is, given an evaluation it is hard to come up with an input to the function on which the function achieves the given evaluation. A one-way function is said to be sub-exponentially strong if it is hard to invert it in sub-exponential time. This ruled out the initial optimism of separating NP from $\mathsf{P/poly}$ using proofs and techniques which are natural. The assumption Razborov and Rudich make, which would be violated if there are "natural" proofs showing super polynomial lower bounds for a circuit class C, is that there are hard (even by sub-exponential time procedures) to invert one way functions in C. It is widely believed that there are sub-exponentially strong one way functions even in small circuit classes like NC^1. A famous example which is believed to be a one way function in $\mathsf{P/poly}$ is the factoring problem, which given two primes outputs their product. The hardness

of inverting this function is the heart of many cryptographic breakthroughs like RSA encryption scheme of [RSA78].

There are two fundamental ways to making progress towards frontier questions in circuit complexity. One is to come up with new combinatorial measures such that there is a potential of proving better lower bounds for certain resources. The other is to modify and strengthen existing measures and/or improve lower bounds against these measures. But given a barrier like "natural" proofs, it is important that the lower bound techniques which we use for these measures are non-natural or at the very least techniques which are not believed to be "natural". In the first part of the thesis we come up with an alternate measure for parameterizing non-monotonicity in Boolean circuits which is tailored towards depth lower bounds. We then prove lower bounds for circuits which have limited non-monotonicity under this measure, using a technique not known to be natural. We also study the associated learning problem for learning circuits which have low monotonicity under this measure, using the properties of the measure we introduce. We could obtain learning algorithms for this problem. However we could not prove an optimality of our result. Towards this goal, we come up with a noise model for studying hardness amplification of learning functions tailored for our measure and outline a strategy using this noise model.

In the second part of the thesis, we study an approach for proving lower bounds for branching programs which is another combinatorial model of computation which captures space in Turing machine. Since polynomial sized branching programs can compute any function in NC^1, the natural proof barrier for P vs NC^1 applies for super polynomial lower bounds against deterministic branching programs also. We consider a method from the 90's devised by Pudlák and Rödl

[PR92] which is also not known to be "natural". The method is based on an algebraic/combinatorial parameter associated with the Boolean function called Projective dimension which the authors show is upper bounded by branching program size. Though their method brings in more tools from algebra and combinatorics to tackle the lower bound problem for branching programs, the best lower bound obtained using their method is only linear in input size. We explain this situation by showing an exponential gap between their measure and branching program size. We remedy this situation by defining a variant of projective dimension which captures branching program size up to polynomial factors.

1.1.2 The depth lower bound for limited non-monotone circuits

In Chapter 3 we extend these depth lower bounds from monotone circuits to limited non-monotone circuits. The amount of non-monotonicity of a circuit is measured by a new measure we introduce called orientation. A generalization of monotone functions is studied under the name *unate functions* (see [IPS97]). We inherit the terminology of *orientation* from that setting. But we remark that our definition is universal unlike the case of unate functions. Unlike the number of negation gates in a circuit, orientation of a circuit is a semantic measure. We first define orientation of a function and extend the definition to circuits. A Boolean function f is said to be of orientation w if there is a circuit computing f with at most w leaf negations (i.e., all the NOT gates are fed by inputs alone). A circuit C is said to be of orientation w, if the Boolean functions computed by every sub-circuit (decided by rooting at an arbitrary internal gate) is of orientation w. Recall that a monotone circuit is a circuit where every sub-circuit computes a monotone function. We first study the structural properties of orientation and prove that it

is a property of the Boolean function f under consideration. And the minimum orientation w of f is achieved only by placing negations on a unique set of w input variables which are decided by the non-monotone hyper edges of the function.

We then proceed to prove interesting properties of "orientation" in terms of circuit size and depth. We show the usefulness of "orientation" by exhibiting a function (non-explicit) which has poly-size log depth circuits computing where only two internal gates compute non-monotone functions. But any monotone circuit computing this function requires super-poly-logarithmic depth.

Our main result which appears in Chapter 3 is super-logarithmic depth lower bounds for **CLIQUE**$(n, n/2)$ and **PMATCH** against circuits of orientation $O(\frac{\sqrt{n}}{\log^{1+\epsilon} n})$. Note that general circuits can have maximum orientation n. The main idea behind our proof is to use the w-orientation circuit C to solve the monotone Karchmer Wigderson game at a cost which is constant multiple of depth d times the orientation w. Since an $O(dw)$ cost monotone Karchmer Wigderson protocol corresponds to a monotone circuit of depth $O(dw)$ [KW88], we get a trade-off between orientation w of a circuit and its depth based on the best monotone depth lower bounds known for the given function. Thus, our method doesn't require a super-polynomial size lower bounds to begin with and works for any function for which super-poly-log depth lower bounds are known.

Since our measure is tailored towards the top-down approach of Karchmer Wigderson games [KW88], it is interesting to see how it interacts with a bottom-up approach of Razborov [Raz85a] and Amano Mauroka [AM05]. In this direction we consider a circuit family computing **CLIQUE**$(n, n^{\frac{1}{6\alpha}})$ with $\ell + k$ negations, where $\ell \leq 1/6 \log\log n$, $\alpha = 2^{\ell+1} - 1$ and at most k negations are computing functions which are sensitive on w inputs with $kw < n/8$ and the remaining ℓ negations

may have arbitrary orientation. We prove such a circuit family must have depth $n^{\frac{1}{2^{\ell+8}}}$. Thus we go beyond the limitation of our method to handle internal gates of orientation $\Omega(\sqrt{n})$ by obtaining super-poly-logarithmic lower bounds when there are only a few negations of "high" orientation. We do this using the specific properties of **CLIQUE**(n, k) and the Amano Maruoka's method [AM05] of proving limited negation lower bounds.

We also consider a setting of orientation where we obtain near matching lower and upper bounds, and where even a slight improvement in the lower bound would answer the NP vs NC question. A circuit is said to be of uniform orientation w if there is a specific subset $R \subseteq [n]$, $|R| = w$ such that every sub-circuit computes a function which can be computed by a circuit which uses leaf negations on variables only from R. In this setting we proved that any circuit which computes **CLIQUE**$(n, n/2)$ and for which there is a subset of vertices S of cardinality $\omega(\log n)$ such that any edge whose both end points are in S does not belong to the uniform orientation set R of the circuit C, must have depth $\omega(\log n)$.

We also proved that such a structural assumption can "almost" be assumed without loss of generality. Specifically we proved that, any circuit C computing **CLIQUE**$(n, n/2)$ of depth $O(\log n)$ can be transformed to another circuit C' of uniform orientation and of depth $O(\log n)$ which has a subset U of vertices which satisfy the above criteria (i.e., none of the edges whose end points in U are in the uniform orientation set R) but U has size only $\theta(\log n)$. Thus, if either our upper bound or lower bound on $|U|$ is improved by a $\log^{\epsilon} n$ factor for any $\epsilon > 0$ we would be able to separate NP from NC.

1.2 Learning problem associated with orientation

Learning theory is the area of theoretical computer science which models and rigorously studies various learning tasks. One of the central themes of learning theory is to learn Boolean functions or, as it is known in the learning community, to learn concepts. The main idea is that there is an unknown function called the concept, say $f : \{0,1\}^n \to \{0,1\}$ that we would like to "learn", using minimal number of queries to f of the which are evaluations of f at various inputs called training data, up to an error parameter ϵ called "accuracy". To learn f up to accuracy ϵ is to come up with the description of another Boolean function called "hypothesis" denoted by h, so that a probability that h agrees with f on an x chosen under the distribution from which the learning algorithm was allowed to query f, is at least $1 - \epsilon$.

Learning monotone functions are of great interest to learning theory. The learning under uniform distribution model has been well studied for monotone functions, where queries are made to f by sampling an x uniformly at random from $\{0,1\}^n$. In a seminal paper, Bshouty and Tamon [BT96] proved that any monotone Boolean function can be learned from uniform random samples to error ϵ in time $n^{O(\sqrt{n}/\epsilon)}$.

Motivated by the robustness of inversion complexity defined by Markov ([Mar58]) and well known lower and upper bounds for learning monotone functions Blais et. al ([BCO+15a]) studied the learning problem for functions computed by circuits with a few negation gates. The proved matching upper and lower bounds for this problem. The learning algorithm for learning functions computed by circuits with a limited number of negation gates is based on an alternate characterization of inversion complexity of f they establish. The matching lower bounds are based

on hardness amplification for learning based on [FLS11].

The measure of non-monotonicity we introduced, orientation, is also a robust measure. This is because weight of minimal orientation of a function is independent of the circuit computing it. Motivated by this and since orientation is also a generalization of monotone functions which have been useful in obtaining depth lower bounds, we study the uniform distribution learning problem for functions which have limited weight of orientation. Based on our characterization of minimal orientation of a Boolean function, we obtain a learning algorithm for functions computed by circuits of limited weight of orientation. Our algorithm is a direct corollary of the learning algorithm of Blais et. al ([BCO$^+$15a]) and our characterization of orientation.

Next we detail why the learning lower bound of [BCO$^+$15a] et. al doesn't work to give learning lower bound for limited orientation functions. Since composing an arbitrary non-monotone function with a monotone function can result in a composed function of arbitrary high weight of orientation we come up with a strategy of hardness amplification similar to that of [FLS11], where we study the composition of a monotone function f with two functions g, h where g is a monotone function and h is function of high weight of orientation. We come up with an information theoretic noise model, similar to that of [O'D04], and show Expected Bias under this noise model as the right parameter for studying the composition for certain family of top functions, like REC-3-MAJORITY. Based on this intuition we also state a conjecture on hardness amplification of learning a specific composition of functions tailored for sparse orientation.

Though we are unable to obtain a learning lower bound for limited orientation functions, we conclude the chapter with important open problems which when

solved, assuming the conjecture, would give such a lower bound but are also of independent interest in Fourier analysis.

1.3 Branching programs and the P vs L problem

In the next part of the thesis, we try the second approach, that is to modify existing combinatorial measures to bring them closer to the computation model. Branching programs are a combinatorial model of computation which abstracts space used in deterministic computation. As discussed earlier one of the central problems of complexity is to understand the relationship between efficient time and efficient space. The efficient time is represented by polynomial time P and efficient space is represented by logarithmic space, L. Though it is known that $\mathsf{L} \subseteq \mathsf{P}$, we do not yet know if $\mathsf{L} \neq \mathsf{P}$.

A combinatorial model of computation which models space in Turing machines is called Branching programs. Unlike circuits which are a "parallel" model of computation branching programs are a "sequential" model of computation. A deterministic branching program is an acyclic graph. An important complexity measure associated with branching programs is its size which is the number of vertices in the branching program. When a space S bounded deterministic Turing machine is converted to a branching program the resulting branching program's is of size $2^{O(S)}$. Because of this connection between logarithm of the size of the branching program and the space used in deterministic Turing machines, to separate P from L it is enough to show a super polynomial lower bound for a function in P against deterministic branching programs.

One of the early lower bounds on branching program size was proved by

Nechiporuk ([Nec66]) in the early 70's. He proved a lower bound of $\Omega(\frac{n^2}{\log^2 n})$ on the deterministic branching programs computing a function which was in L. Proving better branching program size lower bounds is an important problem and hard one at it too. This is best explained by the fact that the best known lower bound on deterministic branching program size, without any additional restrictions like width, is the lower bound of Nechiporuk from the 70's.

1.3.1 Projective dimension

To prove better lower bounds it helps to bring in more combinatorial structure to the of study branching program size. Pudlak and Rodl ([PR92]) in achieved this by introducing a combinatorial/algebraic parameter called Projective dimension which is associated with branching program size.

The linear algebraic parameter they introduced, *projective dimension* (denoted by $\mathsf{pd}_{\mathbb{F}}(f)$, over a field $\mathbb{F}$), is defined on a natural graph associated with the Boolean function f. For a Boolean function $f : \{0,1\}^{2n} \to \{0,1\}$, fix a partition of the input bits into two parts of size n each, and consider the bipartite graph $G_f(U, V, E)$ defined on vertex sets $U = \{0,1\}^n$ and $V = \{0,1\}^n$, as $(u, v) \in E$ if and only if $f(uv) = 1$. The projective dimension of the graph G_f (or any graph in general) denote by $\mathsf{pd}_{\mathbb{F}}(G_f)$, is defined as the smallest d for which there is a vector space W of dimension d (over $\mathbb{F}$) and a function ϕ mapping vertices in U, V to linear subspaces of W such that for all $(u, v) \in U \times V$, $(u, v) \in E$ if and only if $\phi(u) \cap \phi(v) \neq \{0\}$. The strength of this approach is that it bring in more algebraic structure to the pursuit of the lower bound question on branching program size.

Pudlák and Rödl [PR92] showed that if f can be computed by a deterministic branching program of size s, then $\mathsf{pd}_{\mathbb{F}}(f) \leq s$ over any field $\mathbb{F}$. The proof establishes

this connection by producing a subspace assignment using the branching program computing f. Thus, in order to establish size lower bounds against branching programs, it suffices to prove lower bounds for projective dimension of explicit family of Boolean functions.

By a counting argument, Pudlák and Rödl in [PR92] showed that for most Boolean functions $f : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$, $\mathsf{pd}_{\mathbb{R}}(f)$ is $\Omega(\sqrt{\frac{2^n}{n}})$. In a subsequent work, [PR94] also established an upper bound $\mathsf{pd}_{\mathbb{R}}(f) = O(\frac{2^n}{n})$ for all functions. More recently, Rónyai, Babai and Ganapathy [RBG02] established the same lower bound over all fields. Over finite fields $\mathbb{F}$, Pudlák and Rödl [PR92] also showed (by a counting argument) that there exists a Boolean function $f : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ such that $\mathsf{pd}_{\mathbb{F}}(f)$ is $\Omega(\sqrt{2^n})$. However, till date, obtaining an explicit family of Boolean functions achieving such lower bounds has remained elusive.

The best lower bound for projective dimension for an explicit family of functions is for the inequality function (on $2n$ bits, the graph is the bipartite complement of the perfect matching) where a lower bound of ϵn for an absolute constant $\epsilon > 0$ is known [PR92] over $\mathbb{R}$. However, the best known size lower bound that was achieved using this framework is only $\Omega(n)$ which is not better than trivial lower bounds.

1.3.2 Characterizing branching program size using a variant of projective dimension

The approach of Pudlák and Rödl also has the extra advantage that it is also not known to be natural. But using their bridge no one has been able to prove even super-linear lower bounds for branching programs. We address this issue

by proving a stronger connection between branching programs and projective dimension. This is done by defining a variant of projective dimension which captures branching program size up to polynomial factors. Hence any super polynomial lower bounds translates to a super polynomial lower bound for our measure and vice versa. We then continue to show the potential of this approach by showing the best known branching program size lower bound for our variant of projective dimension. This is despite the fact that the equivalence we establish between branching program size and our variant of projective dimension loses a polynomial factor. Also, the lower bound we prove on the branching program size using our variant of projective dimension is only super-linear and not super-polynomial and hence natural proofs barrier do not limit such results.

We also propose a candidate function for proving super-polynomial branching program size lower bounds using this approach.

1.4 Structure of the Thesis

In Chapter 2 we discuss the preliminaries needed for the rest of the thesis. We introduce our new measure of non-monotonicity, orientation, and discuss its properties and associated depth lower bounds in 3. Motivated by the robustness of orientation, we study the learning problem associated with sparsely oriented circuits in Chapter 4. In Chapter 5, we strengthen the approach of Pudlák and Rödl for proving branching program lower bounds using projective dimension by coming up with a variant of projective dimension which characterizes branching program size up to polynomial factors and show the potential of this variant for proving branching program size lower bounds.

CHAPTER 2

Preliminaries

In this chapter, we discuss basic definitions and prerequisites needed for the remaining chapters. In Section 2.1 we define Boolean circuits and related complexity classes. We also discuss limited negation circuits and Karchmer-Wigderson relations in this section. In the next section, Section 2.2 we define the model of branching programs. And then we outline Nechiporuk's sub-function counting technique for proving branching program lower bounds. We finish the section by discussing about graph complexity and projective dimension based approach for proving Branching program lower bounds outlined by Pudlák and Rödl .

2.1 Boolean circuits

A Boolean circuit is a directed acyclic graph (DAG) with a designated root node. Every internal node is labeled with a Boolean function f of its inputs which is either $\wedge, \vee$ or $\neg$ and the edges are called *wires*. Every leaf node is a labeled by an input index. The Boolean function computed by the circuit is the function computed by the root node. Since the leaves have to be labeled by input bits, a circuit is defined for a specific input length. Thus, a family of Boolean functions $\mathcal{F} = \{f_n : \{0,1\}^n \to \{0,1\}\}_{n\in\mathbb{N}}$ is computed by a family of Boolean circuits $C = \{C_n\}_{n\in\mathbb{N}}$, i.e., one circuit for each slice of the Boolean function defined by the input length. The fan-in of a node in a circuit is the in-degree of the node, i.e., the number of nodes which have directed edge from them to the node under consideration.

Similarly, fan-out of a circuit is its out-degree which is the number of nodes the node under consideration is feeding into. A fan-out 1 circuit is a circuit where all internal nodes have fan-out 1, and is called as Formula. Basically formulas are circuits where the underlying DAG is a tree. Formulas represent computations where a partially computed result cannot be reused. A circuit family $C = \{C_n\}_{n\in\mathbb{N}}$ is said to be a bounded fan-in if there is constant $c \in \mathbb{N}$ such that all internal nodes of the circuits in the family have fan-in at most c. The basis of a circuit family C is a family of Boolean functions $\mathcal{B}$ such that all internal nodes are labeled by function from within $\mathcal{B}$. A classical example of a basis is the fan-in 2 basis of 2-bit AND and OR gates and 1-bit NOT gate represented by $\wedge, \vee$ and $\neg$ respectively. This basis is called a universal basis, as any Boolean function can be computed by a circuit family defined over this basis alone. In this thesis, unless explicitly mentioned otherwise, we would be working with bounded fan-in circuits over the fan-in 2, $\wedge, \vee$ and $\neg$ gates.

We also consider De-Morgan circuits in this thesis. A De-Morgan circuit is a circuit where all the $\neg$ gates appear on the leaves of the circuit alone. Any bounded fan-in $\wedge, \vee, \neg$-gate circuit can be converted into a De-Morgan circuit by pushing down the negation gates using De-Morgan laws from Boolean algebra. Recall that De-Morgan laws state that $\neg(\wedge(g,h)) = \vee(\neg g, \neg h))$ and $\neg(\vee(g,h)) = \wedge(\neg g, \neg h))$. Also note that such a transformation does not change the depth of the circuit or formula under consideration whereas the size at most doubles.

2.1.1 Circuit size and depth

The two important complexity resources associated with Boolean circuits are its size and depth. The size of a Boolean circuit is the number of internal nodes in

the circuit. At times size is also defined as the number of *wires* in the circuits and these two definitions are polynomially related for bounded fan-in circuits. But for the rest of the thesis, the size of the circuit would refer to the number of internal nodes in the circuit.

The depth of a circuit, is the length of the longest path from the root of the circuit to any of the leaves. As mentioned earlier this is analogous to the parallel time in the CREW PRAM model of parallel computation ([JÓ92]). The depth of a circuit C is denoted by $\mathbf{Depth}(C)$. For a Boolean function f, $\mathbf{Depth}(f)$ denotes the minimum possible depth of a circuit computing f. By $\mathbf{Depth}_t(f)$ we denote the minimum possible depth of a circuit computing f with at most t negations. Size of a circuit is simply the number of internal gates in the circuit, and is denoted by $\text{size}(C)$. $\text{size}(f), \text{size}_t(f)$ are defined analogous to $\mathbf{Depth}(f), \mathbf{Depth}_t(f)$ respectively. We refer the reader to a standard textbook (cf. [Vol99]) for more details.

In complexity theory, we are interested in the asymptotic growth of these parameters and would be using the famous big-O, Ω, Θ notations to upper and lower bound these parameters. See a standard text book [AB07] on complexity theory for more formal definitions of asymptotic analysis of functions.

The notion of efficient size is polynomial in the input length and the notion of efficient depth is poly-logarithmic in the input size. The class $\mathsf{P/poly}$ is the class of polynomial sized circuits. The class NC represents the class of bounded fan-in, polynomial sized and poly-logarithmic depth circuits. The class NC has finer circuits classes inside it based on the exponent of the logarithm of the depth. These classes are called NC^k where $k \in \mathbb{N}$ and $k > 0$. The class NC^k is defined to be the class of circuits of polynomial size and $O(\log^k n)$ depth , where n is the input size of the circuit. One of the interesting open problems in circuit complexity is

to establish the popular belief that NC^k is different from NC^{k-1}. For any k we do not know this as of now. Since we have defined NC^k for each individual k, an equivalent definition of NC is $\mathsf{NC} = \cup_{k\in\mathbb{N}}\mathsf{NC}^k$.

2.1.2 Monotone circuits and limited negation circuits

Monotone circuits are an important and interesting restriction of general circuits, on which circuit complexity theorists have been able to prove a lot of results. A monotone circuit is a circuit where all the internal gates are either $\wedge$ or $\vee$. That is the circuit does not have $\neg$ gates. Monotone circuits compute only monotone functions. A Boolean function $f : \{0,1\}^n \to \{0,1\}$ is said to be monotone, if for any $x, y \in \{0,1\}^n$, $x \leq y$ implies that $f(x) \leq f(y)$. Note that $\leq$ order on the Boolean Hypercube $\{0,1\}^n$ is defined to be $x \leq y$ iff for all $i \in [n]$, $x_i \leq y_i$. A bit x_i is defined to be less than or equal to bit y_i either if $x_i = y_i$ or $x_i = 0$ and $y_i = 1$. Any monotone function can be computed by a monotone circuit. Equivalently, a function is said to be monotone if and only if it is computed by a monotone circuit.

A Boolean function $f : \{0,1\}^n \to \{0,1\}$ is said to be sensitive on an index $i \in [n]$ if there are inputs $u, v \in \{0,1\}^N$ such that they agree on all indices except i, $u_i \neq v_i$ and $f(u) \neq f(v)$.

We now introduce two important monotone functions we study in this thesis. For a set U, we denote by $\binom{U}{2}$ the set $\{\{u,v\} | u \neq v \in U\}$. In an undirected graph $G = (V, E)$, a clique is a set $S \subseteq V$ such that $\binom{S}{2} \subseteq E(G)$. **CLIQUE**$(n,k)$ is a Boolean function $f : \{0,1\}^{\binom{n}{2}} \to \{0,1\}$ such that for any $x \in \{0,1\}^{\binom{n}{2}}$, $f(x) = 1$ if G_x, the undirected graph represented by the undirected adjacency matrix x has a clique of size k. **CLIQUE**(n,k) is a monotone function as adding edges (equivalent to turning 0 to 1 in adjacency matrix) cannot remove a k-clique, if one already exists.

We denote **CLIQUE**$(n, \frac{n}{2})$ by **CLIQUE**. A perfect matching of an undirected graph $G = (V, E)$ is an $M \subseteq E(G)$ such that no two edges in M share an end vertex and it is such that every vertex $v \in V$ is contained as an end vertex of some edge in M. The corresponding Boolean function **PMATCH** : $\{0,1\}^{\binom{n}{2}} \to \{0,1\}$ is defined as **PMATCH**$(x) = 1$ if G_x contains a perfect matching. Note that **PMATCH** is also a monotone function.

The non-monotonicity in a circuit can be parameterized in various ways. Since syntactic definition of monotone circuits is that they are Boolean circuits without negation gates, one way to parametrize the non-monotonicity is by counting the number of negation gates in a circuit. A circuit family $C = \{C_n\}_{n\in\mathbb{N}}$ is said to be $t : \mathbb{N} \to \mathbb{N}$ negation limited if at most $t(n)$ negation gates appear in C_n.

One natural way to extend the lower bounds against monotone circuits to general circuits is by proving lower bounds for negation limited circuits. Amano and Maruoka ([AM05]) achieves this by proving that any $t(n) = 1/6 \log\log n$ negation limited circuit computing **CLIQUE**$(n, \frac{n}{2})$ requires super polynomial size.

2.1.3 Circuit depth and Karchmer-Wigderson relations

Karchmer and Wigderson ([KW90]) in their 1990's seminal paper introduced a connection between circuit depth and communication complexity. It is a strong connection between circuit depth and communication complexity of a specific two player game where the players say Alice and Bob are given inputs $x \in f^{-1}(1)$ and $y \in f^{-1}(0)$, respectively. In the case of general circuits, the game is denoted by $\mathbf{KW}(f)$ and the goal is to find an index i such that $x_i \neq y_i$. In the case of monotone circuits, the game is denoted by $\mathbf{KW}^+(f)$ and the goal is to find an index i such that $x_i = 1$ and $y_i = 0$. Since monotone circuits compute monotone functions,

$\mathbf{KW}^+(f)$ is defined only for monotone Boolean functions f. We abuse the notation and use $\mathbf{KW}(f)$ and $\mathbf{KW}^+(f)$ to denote communication cost of the best protocol solving the corresponding communication game. The communication cost of a deterministic protocol is the maximum number of bits exchanged on any run of the protocol. Karchmer and Wigderson ([KW88]) proved that for any function f, the best possible depth of any circuit computing f, denoted by $\mathbf{Depth}(f)$ is equal to $\mathbf{KW}(f)$. And for any monotone function f the best possible depth of any monotone circuit computing f, denoted by $\mathbf{Depth}^+(f)$ is equal to $\mathbf{KW}^+(f)$. Raz and Wigderson [RW92] showed that $\mathbf{KW}^+(\mathbf{CLIQUE})$ and $\mathbf{KW}^+(\mathbf{PMATCH})$ are both $\Omega(n)$.

2.2 Branching programs

Deterministic branching programs are a sequential model of computation. A deterministic branching program is an directed acyclic graph with designated "start", "accept" and "reject" nodes. The nodes of the branching program are labeled by input indices and the edges or "wires" coming out a node are labeled by 0 or 1. A given input is said to be accepted by the branching program if there is a path starting from the start node, reaching the accept node, and such that at every node in the path labeled x_i, the wire $b \in \{0, 1\}$ is the next edge if and only in the current input $x_i = b$. Thus, every edge in the deterministic branching program can also be thought of as querying a literal. That is if the edge is labeled 0 and is coming out of a vertex labeled x_i, it can be thought of as that the edge is query the literal $\bar{x_i}$. We would be keeping this dual view, of vertex labeled branching programs whose edges query literals, for the rest of the thesis.

2.2.1 Branching program size

The size of the branching program captures the space used in deterministic Turing machines. Formally, it is the number of vertices in the branching program. Another way to measure the size of branching programs is to measure the number of edges. But since we deal only with deterministic branching programs, the number of edges in the branching program is at most double the size. So we can use either definition of size, barring some constant factors in our results.

2.2.2 Nechiporuk's sub-function counting technique

The lower bound resulting from Nechiporuk's result ([Nec66]) from 1966 is still the best known lower bound against deterministic branching programs. The main idea of his lower bound is something known as the sub-function counting technique. Let f be a function on n variables and $Y_1, \ldots, Y_m$ be a partition of $[n]$ the set of inputs to f. The number of sub functions $c_i(f)$ for an $1 \leq i \leq m$, is the maximum size of a collection S of restrictions $\rho : [n] \setminus Y_i \to 0, 1$ (which leaves only variables in Y_i unset) such that for any two $\rho_1, \rho_2 \in S, \rho_1 \neq \rho_2$, $f_{\rho_1} \not\equiv f_{\rho_2}$. In other words $c_i(f)$ is the number of different function on Y_i that can be obtained from f by setting the variables in $[n] \setminus Y_i$. Nechiporuk proved the following theorem which gives a branching program lower bound based on the sub-function count.

Theorem 2.2.1. *[Nec66] There exists a constant $\epsilon > 0$, such that for any function f depending on all its inputs,*

$$\mathsf{bpsize}(f) \geq \epsilon \sum_{i=1}^{m} \frac{\log c_i(f)}{\log \log c_i(f)}$$

Proof. Given a deterministic branching program computing f, and given a re-

striction $\rho : [n] \setminus Y_i \to 0,1$, one can obtain a deterministic branching program computing f_ρ. This is done by simply removing all the edges which queries literals in $[n] \setminus Y_i$ which are false under ρ. Thus, the edges in the resulting branching program computing f_ρ is at most the number of edges in the original branching program computing f which are query literals from Y_i. Let us call these number of edges s_i. For two different sub-functions f_{ρ_1} and f_{ρ_2} on Y_i, the resulting branching programs obtained from the branching program computing f must be different. Hence we get that the number of branching programs on s_i edges must at least be the number sub-functions, $c_i(f)$.

The proof is completed by an upper bound on the number of deterministic branching programs with at most s_i edges and $|Y_i|$ many variables. Since branching programs are acyclic graphs where edges are labeled by literals an easy calculation (see [Juk12, Proof of Theorem 15.1] for details) shows that the number of different deterministic branching programs is at most $(18|Y_i|s_i)^{s_i}$. Since the function f depends on all the input variables and so does any restriction of f, we get that the number of edges s_i must be at least $|Y_i|$. Thus, the number of branching programs is upper bounded by $2^{O(s_i \log s_i)}$. Together with the fact that this number is at least $c_i(f)$, we get that $s_i = \Omega(\frac{\log c_i(f)}{\log \log c_i(f)})$. The branching program size is at least $\sum_{i=1}^{m} s_i$ as each edge is labeled by exactly one literal and the literals are partitioned according to the partition $Y_1, \ldots, Y_m$ of $[n]$. □

Nechiporuk then used the above theorem with a function which has an exponential number of sub-functions, much larger than the number of branching programs. Nechiporuk used the fact that for a function called Element Distinctness, we have $c_i(ED_n) \geq 2^{\Omega(n)}$ [Juk12, Chapter 1], for a specific partition of the inputs into $O(\frac{n}{\log n})$ parts. The element distinctness function on $n = m(2 \log m)$ bits is de-

fined by partitioning of $[n]$ input indices into m parts obtained by grouping every successive $2\log m$ bits into one part of the partition. Each block of the partition represents a number in $[m^2]$ and the function is 1 if all the numbers represented in binary by all the m parts are different from one another.

Theorem 2.2.2. *[Nec66] The element distinctness function* ED_n *requires deterministic branching programs of size* $\Omega(n^2/\log^2 n)$

Proof. Since $c_i(ED_n) = 2^{\Omega(n)}$, we get that $2^{O(s_i \log s_i)} \geq 2^{\Omega(n)}$, implying that $s_i = \Omega(\frac{n}{\log n})$. Since there are $m = \frac{n}{\log n}$ such s_i and since the branching program size is at least $\sum_{i=1}^{m} s_i$, we get an $\Omega(\frac{n^2}{\log^2 n})$ lower bound on size of deterministic branching programs computing ED_n. □

2.3 A barrier for proving circuit lower bounds : Natural proofs

In this section we will give a brief overview of the Natural proofs barrier and describe what it means for a proof to be natural. It concerns with proofs which try to prove super polynomial lower bounds against a family of circuits C. Razborov and Rudich [RR97] view any proof that a function g doesn't have n^c sized C as a property $\mathcal{P}$ of Boolean functions. Since the proof distinguishes g from all the functions f which have n^c sized C circuits, this property is 1 on g and 0 on all the f's which have n^c sized C-circuits. Such a property $\mathcal{P}$ is said to be n^c-useful against C, or just n^c-useful if $C = \mathsf{P/poly}$. An n^c useful property is said to be "natural" if it additionally satisfies the following properties.

- **Constructiveness** : Given the truth table of a Boolean function $f : \{0,1\}^n \rightarrow \{0,1\}$, there is an algorithm which runs in polynomial (in the size of truth

table) time and computes $\mathcal{P}(f)$. Note that polynomial in truth table size is equivalent to $2^{O(n)}$ time.

- **Largeness** : With probability $1/n$, a function f chosen uniformly at random from set of all the functions from $\{0,1\}^n$ to $\{0,1\}$, satisfies $\mathcal{P}(f) = 1$

[RR97] proved that existence of such a natural proof against $\mathsf{P/poly}$ would contradict the existence of another object called sub-exponentially strong one way functions, which is widely believed to exist. A one way function is a polynomial time computable function $h : \{0,1\}^* \to \{0,1\}^*$ such that for every n and every probabilistic polynomial time algorithm A, the probability given a random $x \in \{0,1\}^n$, A running on $f(x)$ "inverts" $f(x)$, i.e. finds an x' such that $f(x') = f(x)$, is negligible. A sub-exponentially strong one-way function remains hard to invert even if the algorithms are allowed to run in sub-exponential time, i.e. time 2^{n^ϵ}, $\epsilon < 1$. [RR97] proved that, if sub-exponentially strong one-way functions exist then there exists a constant $c \in \mathbb{N}$ such that there is no n^c-useful natural predicate $\mathcal{P}$ useful against $\mathsf{P/poly}$.

Their proof works not just for $\mathsf{P/poly}$ but many circuit classes, as long as they satisfy some nice properties and have a sub-exponentially strong one way functions that are computable in C. For example, it is believed that there are sub-exponentially strong one way functions in even a small circuit class like NC^1. And the [RR97] natural proof theorem would imply that there exists a constant $c \in \mathbb{N}$ such that there is no natural predicate $\mathcal{P}$ which is n^c-useful against NC^1. Hence, even proving super polynomial lower bounds against NC^1 under the assumption of strong one-way functions require non-natural proofs. In their seminal paper, [RR97] also showed that many known exponential lower bounds like the Parity lower bound of [Smo87] and[FSS84] against AC^0 circuits are indeed "natural" proofs.

CHAPTER 3

A new measure of non-monotonicity and associated lower bounds

We study depth lower bounds against non-monotone circuits, parametrized by a new measure of non-monotonicity: the orientation of a function f is the characteristic vector of the minimum sized set of negated variables needed in any DeMorgan circuit (circuits where negations appear only at the leaves) computing f. We prove trade-off results between the depth and the weight/structure of the orientation vectors in any circuit C computing the **CLIQUE** function on an n vertex graph. We prove that if C is of depth d and each gate computes a Boolean function with orientation of weight at most w (in terms of the inputs to C), then $d \times w$ must be $\Omega(n)$. In particular, if the weights are $o(\frac{n}{\log^k n})$, then C must be of depth $\omega(\log^k n)$. We prove a barrier for our general technique. However, using specific properties of the **CLIQUE** function (used in Amano Maruoka (2005)) and the Karchmer–Wigderson framework (Karchmer Wigderson (1988)), we go beyond the limitations and obtain lower bounds when the weight restrictions are less stringent. We then study the depth lower bounds when the structure of the orientation vector is restricted. Asymptotic improvements to our results (in the restricted setting) would separate NP from NC. As our main tool, we generalize Karchmer–Wigderson games (Karchmer Wigderson (1988)) for monotone functions to work for non-monotone circuits parametrized by the weight/structure of the orientation. We also prove structural results about orientation and prove connections between number of negations and weight of orientations required to compute a function.

The results that appear in this chapter are from our works that appear in [KS14] and [KS17].

3.1 Introduction

In this chapter we study depth lower bounds for circuits which are parameterized by their non-monotonicity. Large depth lower bounds are known [KW88, RW92] against monotone circuits. But we do not know any depth lower bounds for limited non-monotone circuits, except for the depth lower bounds implied by size lower bounds on bounded fan-in circuits. Note that a function which requires large (read super logarithmic) depth circuits may not necessarily require a large (read super polynomial) size circuit to compute it. Intuitively, this is because depth captures "parallel time" required to solve the problem and size captures "time" required to solve the problem. In fact showing that "efficient parallel time" is different from "efficient time" is one of the central questions of complexity theory. This is the P versus NC^1 problem where P is all Boolean functions computable in polynomial time on a Turing machine (and hence can be computed by a polynomial size circuit) and NC^1 is the class of Boolean functions which are computed by polynomial size and logarithmic depth circuits. Hence for showing such a separation we need to prove a large depth lower bound for a function which is already computed by polynomial sized circuits. Hence lower bounds of this nature cannot be expected to be proved from a super polynomial size lower bound. For example we know ([RW92]) a $\Omega(\sqrt{(n)})$ lower bound against the **PMATCH** function. But we cannot expect to prove a super polynomial size lower bound on non-monotone circuits computing **PMATCH** as it is already in class P and hence is computed by a polynomial sized circuit. Thus, we need a method of extending the depth lower

bounds against monotone circuits to depth lower bounds for circuits with limited non-monotonicity, but one which is not based on size lower bounds.

Another important issue is how to parametrize non-monotonicity of circuits. A syntactical way of parameterizing non-monotonicity of circuits is to count the number of negation gates employed by the circuit. Amano and Maruoka were successful in using this measure of non-monotonicity to prove super polynomial size lower bounds ([AM05]) against circuit computing the **CLIQUE** function with at most $1/6 \log\log n$ negations. But it is not clear how to use the number of negations as the measure of non-monotonicity to extend depth lower bounds from monotone circuits to limited non-monotone circuits.

We study an alternative way of limiting the non-monotonicity in the circuit. To arrive at our restriction, we define a new measure called *orientation* of a Boolean function. The results we obtain in this chapter are based on our results from [KS14].

A generalization of monotone functions is studied under the name *unate functions*(cf. [IPS97]). We inherit the terminology of *orientation* from that setting. We remark that our definition is universal unlike the case of unate functions.

Definition 3.1.1. A Boolean function $f : \{0,1\}^N \to \{0,1\}$ is said to have *orientation* $\beta \in \{0,1\}^N$ if there is a monotone Boolean function $h : \{0,1\}^{2N} \to \{0,1\}$ such that : $\forall x \in \{0,1\}^N, f(x) = h(x, (x \oplus \beta))$.

If f is a monotone Boolean function, from the above definition it is clear that all-0's vector is an orientation of f. The *weight of an orientation* is simply the number of 1's in β, and can be thought of as a parameter indicating how "close" f is to a monotone function. Note that for any DeMorgan circuit computing f, the characteristic vector of negated input indices forms an orientation of the function f. Indeed, replacing the negated input variables with fresh variables in a DeMorgan

circuit results in a monotone circuit.

The definition can be extended to circuits as well. We consider circuits where the function computed at each gate can be non-monotone. But each gate computes a function whose orientation (with respect to the inputs of the circuits) must be of limited weight. We say a circuit C is weight w oriented if every internal gate of C computes a function which has an orientation β with $|\beta| \leq w$. The weight restriction on a circuit thus defined is a semantic restriction as we are only limiting the weight of orientation of the functions computed at sub-circuits of C. But we do not place any restriction on how (especially in terms of actual negation gates) the functions at sub-circuits are computed in C. We prove the following theorem which presents a depth vs weight trade-off for weight restricted circuits.

Theorem 3.1.2. *If C is a Boolean circuit of depth d and weight of the orientation w ($w > 0$), computing* **CLIQUE** *on an n vertex graph then, $d \times w$ must be $\Omega(n)$.*

In particular, if the weights are $o(\frac{n}{\log^k n})$, the **CLIQUE** function requires $\omega(\log^k n)$ depth. Note that in terms of input size N, this weight is $o(\frac{\sqrt{N}}{\log^k N})$. By contrast, any circuit computing **CLIQUE** has weight of the orientation at each gate at most n^2. We prove the above theorem by extending the Karchmer–Wigderson framework for monotone circuit depth, to the case of non-monotone circuits which are sparsely oriented[1]. The proof depends critically on the route to monotone depth via Karchmer–Wigderson games. This is because it is unclear how to directly simulate weight w-restricted circuit model using a monotone circuit for $w > 0$. We remark that the above theorem applies even to circuits computing **PMATCH**.

[1]In our explanations, we say that a Boolean function is "sparsely oriented" if there is an appropriate upper bound on the weight of the orientation of the function. When it is unrestricted, we describe it as densely oriented. We call a gate sparsely oriented (resp. densely oriented), when the Boolean function computed at the gate is sparsely oriented (resp. densely oriented). In our technical statements we state the weight bounds precisely rather than using this term.

The difficulty in extending the above lower bound to more general lower bounds is the potential presence of gates computing "densely" oriented functions. In this context, we explore the usefulness of gates with non-zero orientation in a circuit. We argue that allowing even a constant number of non-zero (but "dense") weight of orientation gates can make the circuit more powerful in the limited depth setting. In particular, we show (see Theorem 3.3.1) that:

Theorem 3.1.3. *There exists a monotone Boolean function f which cannot be computed by poly-log depth monotone circuits, but there is a poly-log depth circuit computing f such that there are at most two internal gates whose weight of orientation is non-zero.*

We note that the function in Theorem 3.3.1 is derived as a restriction from the non-uniform NC^2 circuit computing **PMATCH** and hence is *not explicit*. The above theorem indicates that the densely oriented gates are indeed useful, and that Theorem 3.1.2 cannot be improved in terms of the number of densely oriented gates it can handle, without using specific properties about the function (for example, **CLIQUE**) being computed.

Going beyond the above limitation, we exploit the known properties of the **CLIQUE** function and use the generalized Karchmer–Wigderson relations to prove lower bounds against circuits with less stringent weight restrictions (in particular, we can restrict the weight restrictions to only negation gates and their inputs)

We prove the following theorem by combining our method with the method of [AM05] for number of negations to obtain the following result.

Theorem 3.1.4. *For any circuit family $C = \{C_m\}$ (where $m = \binom{n}{2}$) computing $\mathbf{CLIQUE}(n, n^{\frac{1}{6\alpha}})$ with $\ell + k$ negation gates, where $\ell \leq 1/6 \log\log n$, $\alpha = 2^{\ell+1} - 1$, at most k negation gates are computing functions which are sensitive only on w inputs[2] with $kw \leq \frac{n}{8}$ and*

[2]i.e., the weight of orientation of the function computed at their input plus orientation of the

the remaining ℓ negations compute functions of arbitrary orientation the follwing holds: $\mathbf{Depth}(C_m) \geq n^{\frac{1}{2^{\ell+8}}}$

This theorem implies that **CLIQUE** cannot be computed by circuits with depth $n^{o(1)}$ even if we allow some constant number of gates to have non-zero (even dense) orientation - thus going beyond the earlier hurdle presented for **PMATCH**. We remark that the above theorem also generalizes the case of circuits with negations at the leaves ($\ell = 0$, and $w = 1$).

It gives hope that by using properties of **CLIQUE** (like hardness of approximation from [AM96] used by [AM05]) one can possibly push the technique further.

We also explore the question of the number of densely oriented gates that are required in an optimal depth circuit. We establish the following connection to the number of negations in the circuit.

Theorem 3.1.5. *For any circuit C with t negations, there is a circuit C' computing the same function such that $size(C') \leq 2^t \times (size(C) + 2^t) + 2^t$, and there are at most $3(2^t - 1)$ internal gates whose orientation is a non-zero vector.*

Next we study circuits where the structure of the orientation is restricted. The restriction is on the number of vertices of the input graph involved in edges indexed by the orientation vector of the function.

Theorem 3.1.6. *If C is a circuit computing the* **CLIQUE** *function on n vertex graph and for each gate g of C, the number of vertices of the input graph involved in edges indexed by β_g (the orientation of gate g) is at most w, then $d \times w$ must be $\Omega(\frac{n}{\log n})$.*

We also study a sub-class of the above circuits for which we prove better lower bounds. A circuit is said to be of *uniform orientation* if there exists a single

function computed at their output is at most w

vector $\beta \in \{0,1\}^N$ such that every gate in it computes a function for which β is an orientation.

Theorem 3.1.7. *Let C be a circuit computing the* **CLIQUE** *function on an n vertex graph, with uniform orientation $\beta \in \{0,1\}^N$ such that there is a subset of vertices U and an $\epsilon > 0$ with $|U| \geq \log^{k+\epsilon} n$ for which $\beta_e = 0$ for all edges e within U, then C must have depth $\omega(\log^k n)$.*

We remark that a DeMorgan circuit has an orientation of weight exactly equal to the number of negated variables. However, this result is incomparable with that of [RW89] against DeMorgan circuits for two reasons : (1) this is for the **CLIQUE** function. (2) the lower bounds and the class of circuits are different.

In contrast to the above theorem, we show that an arbitrary circuit can be transformed into one having our structural restriction on the orientation with $|U| = O(\log^k n)$.

Theorem 3.1.8. *If there is a circuit C computing* **CLIQUE** *with depth d then for any set of $c \log^k n$ vertices U, there is an equivalent circuit C' of depth $d + c \log^k n$ with orientation β such that none of the edges $e(u,v)$, $u, v \in U$ has $\beta_{e(u,v)} = 1$.*

Thus, if either Theorem 3.1.7 is extended to $|U| = \Omega(\log^k n)$ or the transformation in Theorem 3.1.8 can be modified to give $|U| = O(\log^{k+\epsilon} n)$ for some constant $\epsilon > 0$, then a depth lower bound for **CLIQUE** function against general circuits of depth $O(\log^k n)$ will be implied.

The rest of this Chapter is organized as follows. In Section 3.2 we define our new measure, orientation, and prove some interesting properties of orientation. Then in Section 3.3 we show the usefulness of orientation by proving that there is a (non-explicit) function for which the depth required to compute it reduces from

super poly-logarithmic to poly-logarithmic if even two internal gates of non-zero orientation are allowed. In Section 3.4, we show a trade-off between depth of circuits computing **PMATCH**, **CLIQUE** and the weight of orientation of such circuits, the main result of this chapter. In the next section, Section 3.5, we compare the number of negations in the circuits as a measure of non-monotonicity with weight of orientation as a measure of non-monotonicity. In Section 3.6, we show how to combine Amano and Maruoka's negation limited lower bounds with our orientation based lower bound. In Section 3.7 we impose some structural restrictions on orientation and prove improved lower bounds. We then generalize De-Morgan circuits using a notion of "uniform" orientation in Section 3.8. In Section 3.9 we show how to prove an NC lower bound for functions like **PMATCH** and **CLIQUE** assuming a structure theorem related to Orientation. In Section 3.10 we prove a structure theorem which is "almost" the assumed structure theorem which would give NC lower bounds. We conclude the chapter by discussing the naturalness of our approach in Section 3.11.

3.2 Properties of orientation

In this section, we systematically study and include observations about the new measure of non-monotonicity that we introduced, orientation. We start by recalling the definition of orientation :

Definition 3.2.1 (Orientation of a Boolean function)**.** a function $f : \{0,1\}^N \to \{0,1\}$ is said to have *orientation* $\beta \in \{0,1\}^N$ if there is a monotone function $h : \{0,1\}^{2N} \to \{0,1\}$ such that $\forall x \in \{0,1\}^N$, :

$$f(x) = h(x, (x \oplus \beta))$$

We first argue that for any function $f(x)$ there is a vector β and a monotone function h that satisfies the conditions of above definition. Let C be any circuit computing f. Convert C into a DeMorgan circuit C' by pushing down the negations via repeated applications of De-Morgan's law. In C', for every $i \in [N]$ replace $\bar{x_i}$ with a new variable y_i. Thus, C' on inputs x, y is computes a monotone function. Since there are N input variables at most N y_i's are needed. Let $h = C'(x_1, \ldots, x_N, y_1, \ldots, y_N)$ be the monotone function computed by C' after replacing the negated inputs by fresh variables. Clearly h satisfies the required form with β defined as $\beta_i = 1$ if and only if $\bar{x_i}$ appears in C'.

Property 3.2.2 (**Upward Closure**). *If $\beta \in \{0,1\}^N$ is an orientation for a function f, then any $\beta' \geq \beta$ is also an orientation for f by definition of orientation.*

Proof. To see this, let h_β be the monotone function guaranteed by definition of orientation for orientation β. The monotone function $h_{\beta'}$ can be defined to be h_β itself. □

We also demonstrate this through an example. Consider the Boolean function $f(x_1, x_2, x_3) = x_1 \vee \bar{x_2}\bar{x_3}$. Vector $\beta = 011$ is a valid orientation for f because we have the monotone function $h(x_1, x_2, x_3, y_1, y_2, y_3) = x_1 \vee y_2 y_3$ which behaves as f when $y_1 = x_1 \oplus 0$, $y_2 = x_2 \oplus 1$, $y_3 = x_3 \oplus 1$. Note that vector $\beta' = 111$ is also a valid orientation vector because we can use $h' = h$ as h ignores y_1.

Using Property 3.2.2, we prove a necessary condition for vectors to be valid orientation of a Boolean function f, which we state below.

Property 3.2.3 (**Necessary Conditions for Orientation**). *Let $\beta \in \{0,1\}^N$ be an orientation for a function f. If there exists a pair (u, v) such that $u_i = 0, v_i = 1$, $u_{[N]\setminus\{i\}} = v_{[N]\setminus\{i\}}$ and $f(u) = 1, f(v) = 0$ then, $\beta_i = 1$.*

Proof. Let h be the monotone function corresponding to f for β such that $\forall x, f(x) = h(x, x \oplus \beta)$. Assume to the contrary that $\beta_i = 0$. Since $u_{[N]\setminus\{i\}} = v_{[N]\setminus\{i\}}$, we have that $u_{[N]\setminus\{i\}} \oplus \beta = v_{[N]\setminus\{i\}} \oplus \beta$ for any β. Hence $(u, u \oplus \beta), (v, v \oplus \beta)$ differs only in two indices, namely $i, N + i$. At i, $u_i = 0, v_i = 1$. Since $\beta_i = 0$, $u_{N+i} = u_i = 0, v_{N+i} = v_i = 1$. Hence we get that $(u, u \oplus \beta) < (v, v \oplus \beta)$, but $h(u, u \oplus \beta) = 1, h(v, v \oplus \beta) = 0$, a contradiction to monotonicity of h. □

It is not a priori clear that a minimal (with respect to $<$ relation on the Boolean hypercube $\{0,1\}^N$) orientation for a function f is unique. We prove that it is indeed unique using the previous property.

Property 3.2.4 (**Minimal Orientation is Unique**). *Minimal orientation for a function $f : \{0,1\}^N \to \{0,1\}$ is unique and it is $\beta \in \{0,1\}^N$ such that $\beta_i = 1$ if and only if there exists a pair (u, v) such that $u_i = 0, v_i = 1$, $u_{[N]\setminus\{i\}} = v_{[N]\setminus\{i\}}$ and $f(u) = 1, f(v) = 0$.*

Proof. From Property 3.2.3, it is clear that any orientation β' of a function f is such that $\beta \leq \beta'$. We claim that negations of variables in β suffices to compute f using a DeMorgan circuit. Define a partial function $h : \{0,1\}^{2N} \to \{0,1\}$ associated with orientation β of f as $h(x, x \oplus \beta) \triangleq f(x)$. We claim that this partial function has an extension which is a monotone function. We claim that for any $u, v \in \{0,1\}^N$ such that $u \leq v$ and $f(u) = 1, f(v) = 0$, there exists an $i \in [N]$ such that $u_i = 0, v_i = 1$ and $\beta_i = 1$. Let $w_0 = u \leq w_1 \leq \cdots \leq w_j \leq w_{j+1} \leq \cdots \leq w_k = v$ be a chain on the hypercube between u and v. Take the minimum j such that $f(w_j) = 1$ and $f(w_{j+1}) = 0$. Since w_j, w_{j+1} satisfies assumptions of Property 3.2.3, for the i where w_j and w_{j+1} differs, $\beta_i = 1$. Since $u \leq w_j$ and the ith bit of w_j is 0, we get $u_i = 0$. Similarly $v_i = 1$ as $v \geq w_{j+1}$ and the ith bit of w_{j+1} is 1. With this claim we can prove that for any $(s, s \oplus \beta)$ and $(t, t \oplus \beta)$ either they are incomparable or $f(s) \geq f(t)$ if and only if $(s, s \oplus \beta) \geq (t, t \oplus \beta)$. Suppose that $f(s) < f(t)$ and $(s, s \oplus \beta) \geq (t, t \oplus \beta)$.

Since $(s, s \oplus \beta) \geq (t, t \oplus \beta)$, $s \geq t$ and $f(s) = 0, f(t) = 1$ as $f(s) < f(t)$. But then we are guaranteed by the earlier claim an $i \in [N]$ such that $s_i = 1, t_i = 0, \beta_i = 1$. Since $\beta_i = 1$, $s_i \oplus \beta_i = 0$ and $t_i \oplus \beta_i = 1$ whereas $s_i = 1, t_i = 0$ implying that $(s, s \oplus \beta) \not\geq (t, t \oplus \beta)$, a contradiction. Thus, the partial function we defined will never have a chain with a 1 to 0 transition. Also any partial function h which does not have a $1 \to 0$ transition on any of the chains of the Boolean hypercube, has an extension to a function which is monotone. □

With Property 3.2.4 characterizing unique minimal orientation, we can illustrate how weight of *orientation* captures *non-monotonicity* of Boolean functions. By Property 3.2.4 a Boolean function is monotone if and only if the weight of minimal orientation is 0. On the other hand, a highly non-monotone function[3], like $\textsc{Parity}(x_1, \ldots, x_N) = \oplus_{i=1}^{N} x_i$ has weight of minimal orientation the maximum possible, N. This follows from Property 3.2.4, as for any $i \in [N], i \neq 1$ consider $u = 10^{N-1}$ and $v = 10^{i-2}10^{N-i+1}$. Note that $u \leq v$, u, v differ only at index i, and $\textsc{Parity}(u) = 1$ but $\textsc{Parity}(v) = 0$. Thus implying that for any $i \in N, i \neq 1$, the minimal orientation β of $\textsc{Parity}$ has $\beta_i = 1$. For $i = 1$, consider $u = 0^{N-1}1$ and $v = 10^{N-2}1$.

3.3 High orientation provably (non-explicit) reduces depth

We show that even as few as two "densely" oriented internal gates can help to reduce the depth from super poly-log to poly-log for some functions.

Theorem 3.3.1. *There exists a monotone Boolean function f such that it cannot be com-*

[3] $\textsc{Parity}$ requires $\log N$ negation gates in any circuit computing it [Mar58], and this is the maximum number of negations needed for any function [Fis75].

puted by poly-log depth monotone circuits, but there is a poly-log depth circuit computing it with at most two internal gates having non-zero orientation β.

Proof. It is known [RW92] that **PMATCH** does not have monotone circuits of poly-log depth. But if arbitrary negations are allowed then there is an $O(\log^2 N)$ depth circuit computing **PMATCH** [Lov79]. The monotone function f claimed in the theorem is obtained from poly-log depth circuit C computing **PMATCH**. Fischer's theorem guarantees that without loss of generality we can assume that C has at most $\log N$ negations.

If there is a poly-log depth circuit having exactly one negation computing **PMATCH**, then Theorem 3.1.5 can be applied to get a circuit of poly-log depth having at most two gates of non-zero orientation. Otherwise, the circuit has $t \geq 2$ negations, and there is no poly-log depth circuit computing the same function with one negation. Let g_1 denote the input to the first negation gate(in the topological sorted order) in C. From C obtain C' by replacing g_1 with a new variable, say y_1. Let C'_0, C'_1 denote the circuits obtained by setting y_1 to 0,1 respectively. The corresponding functions f_0, f_1 need not be monotone. Hence we define monotone functions f'_0, f'_1 from f_1, f_0 :

$$\begin{aligned} f'_0(x) &= f_0(x) \vee g_1(x) \\ f'_1(x) &= f_1(x) \wedge g_1(x) \end{aligned}$$

When $g_1(x) = 0$, $f_0(x) = f(x)$ and when $g_1(x) = 1$, $f'_0(x) = 1$. Hence f'_0 is monotone. A similar argument can be used to establish that f'_1 is monotone. Note that both f'_0, f'_1 have poly-log depth circuits with at most $t - 1$ negation gates.

We claim that one of f'_0, f'_1 does not have a monotone circuit of poly-log depth.

Otherwise from poly-log depth monotone circuits computing f_0', f_1' and the monotone circuit of poly-log depth computing g_1 we can get a poly-log depth circuit computing f with one negation : use $\overline{g_1}(x)$ as a selector to select $f_1'(x)$ or $f_0'(x)$ as which is appropriate. This circuit computes f because, by definition, $(g_1(x) \wedge f_1'(x)) \vee (\overline{g_1}(x) \wedge f_0'(x)) = f(x)$. This contradicts our assumption that there is no circuit of poly-log depth computing f with one negation.

Applying the procedure once, we get out of either $f_0^{'}$ or $f_1^{'}$, a monotone function f' which has a $t-1$ negation poly-log depth circuit, but it has no monotone circuit of poly-log depth computing it. If the function f' has a poly-log depth circuit with one negation then Theorem 3.1.5 can be applied to get the desired function. Otherwise apply the procedure on f' as f' is a monotone function which does not have any poly-log depth circuit with at most one negation. Applying the procedure at most t ($t \leq \log N$) times we get to a monotone function f' having a poly-log depth circuit with one negation, but has no monotone poly-log depth circuit. Applying Theorem 3.1.5 on the one negation circuit gives a poly-log depth circuit with at most two gates of non-zero orientation. □

This theorem combined with the "sparse" orientation protocol implies that the two non-zero orientations β_1, β_2 are such that $|\beta_1| + |\beta_2|$ is not only non-zero but is super poly-log. Since our protocol spends $|\beta_1| + |\beta_2|$ for handling these two gates, and on the remaining gates in the circuit it spends 1 bit each, the cost of the sparse orientation protocol will be at most $|\beta_1| + |\beta_2| + \mathbf{Depth}(C)$. Thus $|\beta_1| + |\beta_2|$ is at least $\mathbf{KW}^+(f) - \mathbf{Depth}(C)$ which is super poly-log as $\mathbf{Depth}(C)$ is poly-log and $\mathbf{KW}^+(f)$ is super poly-log.

Remark 3.3.2. By Theorem 3.3.1 we get a function which has an NC^2 circuit with two non-zero orientation gates which has no monotone circuit of poly-log depth.

Thus, our bounds cannot be strengthened to handle higher weights without incorporating the specifics of the function being computed. In Section 3.7, we rescue the situation slightly using the specific properties of the **CLIQUE** function.

Remark 3.3.3. The proof of Theorem 3.3.1 also implies that there is a monotone function f (not explicit) such that there is a one negation circuit in NC^2 computing it, but any monotone circuit computing f requires super-poly-log depth.

3.4 Depth lower bounds for circuits of sparse orientation

In this section, we prove Theorem 3.1.2 which shows the trade-off between depth and weight of orientation of the internal gates of a circuit. We prove the following lemma which is the main result of this chapter. Note that the Theorem 3.1.2 immediately follows from the following lemma as we know [RW92] that monotone depth of the **CLIQUE** function is $\Omega(n)$.

Lemma 3.4.1. If C is a depth d circuit computing a monotone Boolean function $f : \{0,1\}^N \to \{0,1\}$ which is sensitive on all its indices and each internal gate of C computes a Boolean function whose orientation has weight at most w, then $d \times (4w + 1) \geq \mathbf{KW}^+(f)$.

Proof. The proof idea is to devise a protocol for $\mathbf{KW}^+(f)$ using C having $\mathbf{Depth}(C)$ rounds and each round having a communication cost of $4w + 1$.

Alice is given $x \in f^{-1}(1)$ and Bob is given $y \in f^{-1}(0)$. The goal is to find an index i such that $x_i = 1, y_i = 0$. For simplicity of presentation we would assume that the $\wedge, \vee$ gates are of fan-in 2 and $\neg$ is of fan-in 1. It is easy to see the proofs for constant

fan-in from the current proof. The protocol is described in Algorithm 1.

Algorithm 1 Modified Karchmer–Wigderson Protocol

{Let x' and y' be the current inputs. At the current gate g computing f, with the input gates g_1 and g_2 where f_1 and f_2 are the functions computed, let β_1, β_2 be the corresponding orientations (and are known to both Alice and Bob). If g_1 or g_2 is a negation gate, let γ_1 and γ_2 be the orientation vectors of input functions to g_1 and g_2, otherwise they are 0-vectors. Let $\alpha = \beta_1 \vee \beta_2 \vee \gamma_1 \vee \gamma_2$. Let $S = \{i : \alpha_i = 1\}$, x_S is the substring of x indexed by S.}
if g is $\wedge$ **then**
  Alice sends x'_S to Bob. Bob compares x'_S with y'_S.
  if there is an index $i \in S$ such that $x'_i = 1$ and $y'_i = 0$ **then**
    Output i.
  else
    Define $y'' \in \{0,1\}^N$: $y''_S = x'_S$ and $y''_{[N]\setminus S} = y'_{[N]\setminus S}$.
    Bob sends $j \in \{1,2\}$ such that $f_j(y'') = 0$ to Alice. They recursively run the protocol on g_j with $x' = x'$ and $y' = y''$.
  end if
end if
if g is $\vee$ **then**
  Bob sends y'_S to Alice. Alice compares y'_S with x'_S.
  if there is an index $i \in S$ such that $x'_i = 1$ and $y'_i = 0$ **then**
    Output i.
  else
    Define $x'' \in \{0,1\}^N$: $x''_S = y'_S$ and $x''_{[N]\setminus S} = x'_{[N]\setminus S}$.
    Alice sends $j \in \{1,2\}$ such that $f_j(x'') = 1$ to Bob. They recursively run the protocol on g_j with $x' = x''$ and $y' = y'$.
  end if
end if

We now prove that the protocol (Algorithm 1) solves $\mathbf{KW}^+(f)$. The following invariant is maintained during the run of the protocol and is crucial for the proof of correctness.

Invariant: When the protocol is at a node which computes a function f with orientation vector β it is guaranteed a priori that the inputs held by Alice and Bob, x' and y' are equal on the indices where $\beta_i = 1$, $f(x') = 1, f(y') = 0$ and the restriction of f obtained by fixing variables where $\beta_i = 1$ to $x'_i (= y'_i)$ is a monotone function.

If the invariant is maintained, we claim that when the protocol stops at a leaf

node of the circuit computing a function f with $f(x') = 1$ and $f(y') = 0$ then $f = x_i$ for some $i \in [N]$. If the leaf node is a negative literal, say $\bar{x_i}$ then by Property 3.2.3, orientation of $\bar{x_i}$ has $\beta_i = 1$. By the guarantee that $x'_\beta = y'_\beta$, we have $x'_i = y'_i$, contradicting $f(x') \neq f(y')$. Hence whenever the protocol stops at a leaf node it is guaranteed that the leaf is labeled by a positive literal. And when the input node is labeled by a positive literal x_i, then a valid solution is output, as $f(x') = 1, f(y') = 0$ implies $x'_i = 1$ and $y'_i = 0$. Note that during the run of the protocol we only changed x, y at indices i where $x_i \neq y_i$, to $x'_i = y'_i$. Hence, for any index where $x'_i \neq y'_i$ it is the case that $x_i = x'_i$ and $y_i = y'_i$.

Now to prove the invariant note that it is vacuously true at the root gate as f is a monotone function implying $\beta = 0^N$, and in the standard $\mathbf{KW}^+(f)$ game $x \in f^{-1}(1)$ and $y \in f^{-1}(0)$. We argue that, while descending down to one of the children of the current node the invariant is maintained. To begin with, we show that the protocol does not get stuck in step 8 (and similarly for step 17). To prove this, we claim that at an $\wedge$ gate $f = f_1 \wedge f_2$, if the protocol failed to find an i in step 4 such that $x'_i = 1, y'_i = 0$ then on the modified input y'' at least one of $f_1(y'')$ or $f_2(y'')$ is guaranteed to be zero. Since the protocol failed to output an i such that $x'_i = 1, y'_i = 0$, it must be the case that $x'_i \leq y'_i$ for indices indexed by β_1, β_2. Let U be the subset of indices indexed by β_1 and β_2 where $x_i = 0$ and $y_i = 1$. Bob obtains y'' from y' by setting $y''_i = 0$ for all $i \in U$. Thus, we have made sure that x' and y'' are the same on the variables whose negations are required to compute f, f_1 and f_2.

Consider the functions $f', f'' : \{0,1\}^{N-|\beta_1 \vee \beta_2|} \to \{0,1\}$ which are obtained by restricting the variables indexed by orientation vectors of f_1 and f_2 to the value of those variables in x'. Both f' and f'' are monotone as they are obtained by restricting all negated input variables of the DeMorgan circuits computing f_1 and

f_2 for orientations β_1 and β_2 respectively. The changes made to x', y' were only at places where they differed. Thus, at all the indices where x', y' were the same, x', y'' is also same. Hence the monotone restriction $f_{x'_\beta}$ of f obtained by setting variables indexed by β to their values in x' is a consistent restriction for y'' also. Note that $y'' \leq y'$. Hence $f(y'') = 0$ because y'' agrees with y' on variables indexed by β (as x'' agrees with y' and y'' on variables indexed by β) implying $f_{x'_\beta}(y''_{[N]\setminus\beta}) \leq f_{x'_\beta}(y'_{[N]\setminus\beta}) = 0$. Since $f(y'') = 0$, it is guaranteed that one of $f_1(y''), f_2(y'')$ is equal to 0. Bob sets $y' = y''$ and sends 0 if it is $f_1(y'') = 0$ or 1 otherwise, indicating Alice which node to descend to. Note that $x'_{\beta_1} = y''_{\beta_1}, x'_{\beta_2} = y''_{\beta_2}$ and the restrictions of f_1, f_2 to $x'_{\beta_1}, x'_{\beta_2}$ respectively gives monotone functions f', f'' thus maintaining the invariant for both f_1 and f_2.

We claim that if any of the input gates g_1, g_2 to the current $\wedge$ gate g is a $\neg$ gate then the protocol will not take the path through the negation gate. To argue this, we use the following lemma.

Lemma 3.4.2. If ℓ and complement of ℓ are functions with orientations β, γ, then for all $x, y \in \{0,1\}^N$ such that $x_{\beta\vee\gamma} = y_{\beta\vee\gamma}$, $\ell(x) = \ell(y)$.

Proof. We know that for a function ℓ, if there exists a pair $(u, v) \in \{0,1\}^N \times \{0,1\}^N$ with $u \leq v$, $u_i \neq v_i$, $u_{[N]\setminus\{i\}} = v_{[N]\setminus\{i\}}$ and $\ell(u) = 1, \ell(v) = 0$ then by Property 3.2.3 for every orientation $\beta, \beta_i = 1$. Let i be an index on which ℓ is sensitive, i.e., there exists $(u, v) \in \{0,1\}^N \times \{0,1\}^N$ with $u \leq v$, $u_i \neq v_i$, $u_{[N]\setminus\{i\}} = v_{[N]\setminus\{i\}}$ and $\ell(u) \neq \ell(v)$. Note that ℓ is sensitive on i need not force $\beta_i = 1$, as it could be that $\ell(u) = 0$ and $\ell(v) = 1$. But in this case $\bar{\ell}(u) = 1$ and $\bar{\ell}(v) = 0$, hence $\gamma_i = 1$ for $\bar{\ell}$. Hence, ℓ is sensitive only on indices in $\beta \vee \gamma$. □

The lemma establishes that every negation gate in a weight w oriented circuit

computes a function which is sensitive on at most $2w$ indices. Hence if $2w < N$ the root gate cannot be a negation gate for a function sensitive on all inputs. Suppose at a node one of the children is a negation gate, say f_1. Since we ensure $x'_{\beta_1 \vee \gamma_1} = y''_{\beta_1 \vee \gamma_1}$, Lemma 3.4.2 implies $f_1(x') = f_1(y'')$. But the protocol does not descend down a path where x', y'' are not separated. Hence the claim.

This also proves that when the protocol reaches an $\wedge$ node where both children are negation gates, at the round for that node the protocol outputs an index i and stops. Otherwise, since we ensure $x'_S = y''_S$, $f_1(y'') = f_1(x') = 1$ and $f_2(y'') = f_2(x') = 1$ by Lemma 3.4.2. But this contradicts the fact that at a node $f = f_1 \wedge f_2$ either $f_1(y'') = 0$ or $f_2(y'') = 0$ (or both).

The proof of equivalent claims for an $\vee$ gate is similar except for the fact that Alice modifies her input.

Thus, using the above protocol we are guaranteed to solve $\mathbf{KW}^+(f)$. Communication cost of any round is at most $4w+1$. Because if any of the children is a negation gate then we have to send its orientation along with the orientation of its complement. The protocol clearly stops after $\mathbf{Depth}(C)$ many rounds. Thus, the communication complexity of the protocol is upper bounded by $\mathbf{Depth}(C) \times (4w + 1)$. □

3.5 Comparison of orientation to negations

Since the weight of the orientation can be thought of as a measure of non-monotonicity in a circuit, a natural question to explore is the connection between the number of negation gates and number of non-zero orientation gates required to compute a function f. We now prove Theorem 3.1.5 from the introduction of this Chapter.

Proof. In C_N replace input of each negation by new a variable, say $y_1, \ldots, y_t$, thus obtaining a circuit $C''_N(x_1, \ldots, x_N, y_1, \ldots, y_t)$. Let $g_1, \ldots, g_t$ be the inputs to the t negation gates (in topologically sorted order) of C_N. Note that for each setting of $y_1, \ldots, y_t$ to some $b \in \{0,1\}^t$, $C''_N(x, b)$ is a monotone circuit computing a monotone function on $x_1, \ldots, x_N$. Hence the weight of orientation of each internal gate in $C''_N(x, b)$ is zero. Let $g_{i,b}$ for $i \in [t], b \in \{0,1\}^t$ denote the monotone function computed by the sub-circuit C_{g_i} of C_N rooted at gate g_i, where $g_1, \ldots, g_{i-1}$ are set to $b_1, \ldots, b_{i-1}$ respectively. Thus, we can write f as:

$$f(x_1, \ldots, x_N) = \bigvee_{b \in \{0,1\}^t} \left(\bigwedge_{i=1}^{t} g_{i,b}^{b_i}(x) \right) C''_N(x, b), \tag{3.1}$$

where g^0 denotes $\overline{g}$ and g^1 denotes g. When $t = 1$, then the above expression becomes $f(x) = g(x)C(x,1) \vee \overline{g}(x)C(x,0)$. In this case the only gates which can have non-zero orientation are the negation computing $\overline{g}$, $\wedge$ computing $\overline{g}(x)C(x,0)$ and the root gate (if the function computed is non-monotone). Hence when $t = 1$ the circuit has at most three gates with non-zero orientation if the circuit computes a non-monotone function and at most two gates of non-zero orientation otherwise.

Consider the formulation of a circuit C' computing f given in Equation (3.1). Clearly $\text{size}(C') \leq 2^t \times (\text{size}(C_N) + 2^t) + 2^t$. All internal gates in $C''_N(a, b)$ are monotone. Non-zero orientation is needed only for computing:

- $\bigwedge_{i \in [t], b_i = 0} \overline{g_{i,b}}$
- $\wedge$ of $\bigwedge_{i \in [t], b_i = 0} \overline{g_{i,b}}$ with $\bigwedge_{i \in [t], b_i = 1} g_{i,b} \wedge C''(x, b)$
- the $\vee$-tree, computing $\sum$ of 2^t terms which are potentially of non-zero orientation.

Instead of computing $\bigwedge_{i \in [t], b_i = 0} \overline{g_{i,b}}$, we can compute the equivalent formula $\overline{\bigvee_{i \in [t], b_i = 0} g_{i,b}}$.

For computing $\overline{\bigvee_{i\in[t],b_i=0} g_{i,b}}$ we just need one negation gate as all $g_{i,b}$ are monotone circuits . To compute the $\wedge$ of this intermediate product with $\bigwedge_{i\in[t],b_i=1} g_{i,b} \wedge C''(x,b)$ one more gate is needed. Thus, the total number of gates needed is 2 for each b except for $b = 1^t$ in which case we don't need any negations. Let us call the number of such gates K_1. By the above analysis, $K_1 = \sum_{b\in\{0,1\}^t, b\neq 1^t} 2 = 2(2^t - 1)$. The remaining gates are the internal gates in the $\vee$ tree implementing the sum of terms. Since there are 2^t leaves, the number of internal nodes in the tree, say K_2 is at most $2^t - 1$. Hence the total number of nodes with non-zero orientation is at most $K_1 + K_2 = 3(2^t - 1)$. □

Remark 3.5.1. In conjunction with the result of Fischer [Fis75], this implies that it is enough to prove lower bounds against circuits with at most $O(N)$ internal nodes of dense orientations, to obtain lower bounds against the general circuits.

3.6 A new depth lower bound combining orientation and negations

The number of gates with high orientations can be arbitrary in general. In this subsection we give a proof for Theorem 3.1.4. We first extend our technique to handle the low weight negations efficiently so that we get a circuit on high weight negations (see Lemma 3.6.1 below). To complete the proof of Theorem 3.1.4, we appeal to depth lower bounds against negation-limited circuits computing **CLIQUE**$(n, n^{\frac{1}{6}\alpha})$.

Lemma 3.6.1. For any circuit family $C = \{C_N\}$ computing **CLIQUE**$(n, n^{\frac{1}{6}\alpha})$ where there are k negations in C_N computing functions which are sensitive only on $2w$ input bits (i.e., the orientation of their input as well as their output is at most

w) with $kw \leq \frac{n}{8}$ and the remaining ℓ negations compute functions of arbitrary orientation: $\mathbf{Depth}(C_N) \geq \mathbf{Depth}_\ell(\mathbf{CLIQUE}(\frac{3n}{4}, n^{\frac{1}{6}\alpha}))$

Proof. Since k negations of C_N depend only on kw inputs (i.e, edges) the number of vertices in the graph which have at least one of its edges indexed by one of the k negations is $2kw$. Let this set of vertices be S. Then $|S| \leq \frac{n}{4}$. In C_N set input variables corresponding to edges in $\binom{S}{2}$ and the variables corresponding to edges between S and $[N] \setminus S$ to 0. Note that the circuit C'_N obtained from C_N by this restriction computes $\mathbf{CLIQUE}(\frac{3n}{4}, n^{\frac{1}{6}\alpha})$. Note that all the k negations which are sensitive only on edges indexed by $\binom{S}{2}$ is fixed to a constant as $\binom{S}{2}$ is fixed. Hence C'_N has at most ℓ negations. Hence the theorem.

□

By a straightforward application of the technique used in [AM05] to prove size lower bounds against circuits with limited negations computing $\mathbf{CLIQUE}(n, n^{\frac{1}{6}\alpha})$ we obtain the size version of following lemma

Lemma 3.6.2. For any circuit C computing $\mathbf{CLIQUE}(n, n^{\frac{1}{6\alpha}})$ with ℓ negations where $\ell \leq 1/6 \log\log n$ and $\alpha = 2^{\ell+1} - 1$,

$$\mathbf{Depth}_\ell(f) \quad \geq \quad n^{\frac{1}{81\alpha}}$$

Combining Lemma 3.6.1 and Lemma 3.6.2 completes the proof of Theorem 3.1.4.

Since the trade-off result stated in Lemma 3.6.2 is not explicitly stated and proved in [AM05], we present the relevant part of the proof technique in [AM05] with careful choice of parameters obtaining the trade-off. For consistency with notation used in [AM05], *for the remainder of this section we will be denoting the*

number of vertices in the graph by m.

The main idea in [AM05] is to consider the boundary graph of a function f, defined as $G_f = \{(u,v)|\Delta(u,v) = 1, f(u) \neq f(v)\}$ where $\Delta(u,v)$ is the hamming distance. They prove that if there is a t negations circuit C computing f then the boundary graph f must be covered by a union of boundary graphs of 2^{t+1} functions obtained by replacing the negations in C by variables and considering the input functions of t negation gates and the output gate where the negations in the sub-circuit considered are restricted to constants.

They prove that,

Lemma 3.6.3. [AM05, Theorem 3.2] Let f be a monotone function on n variables. For any positive integer t,

$$\text{size}_t(f) \geq \min_{F'=\{f_1,\ldots,f_\alpha\}\subseteq \mathcal{M}^n} \left\{ \max_{f'\in F'} \{size_{mon}(f')\} \mid \bigcup_{f'\in F'} G(f') \supseteq G(f) \right\}$$

where $\alpha = 2^{t+1} - 1$ and $G(f')$ denotes the boundary graph of the function f'.

The size lower bound they derive crucially depends on the following lemma which states that no circuit of "small" size can "approximate" clique in the sense that either it rejects all the "good" graphs or accepts a huge fraction of "bad" graphs.

Lemma 3.6.4. [AM05, Theorem 4.1] Let s_1, s_2 be positive integers such that $64 \leq s_1 \leq s_2$ and $s_1^{1/3} s_2 \leq \frac{m}{200}$. Suppose that C is a monotone circuit and that the fraction of good graphs in $I(m, s_2)$ such that C outputs 1 is at least $h = h(s_2)$. Then at least one of the following holds:

- The number of gates in C is at least $(h/2)2^{s^{1/3}/4}$.

- The fraction of bad graphs in $O(m, s_1)$ such that C outputs 0 is at most $2/s_1^{1/3}$.

where a "good" graph in $I(m, s_2)$ is a clique of size s_2 on m vertices and no other edges and a "bad" graph in $O(m, s_1)$ is an $(s_1 - 1)$-partite graph where except for at most one partition the partitions are balanced and of size $\lceil \frac{m}{s_1 - 1} \rceil$ each.

Lemma 3.6.5. For any circuit C computing **CLIQUE**$(m, m^{\frac{1}{6\alpha}})$ with t negations with $t \leq 1/6 \log\log m$, size of C is at least $2^{m^{\frac{1}{81\alpha}}}$ where $\alpha = 2^{t+1} - 1$.

Proof. The proof is similar to the proof ([AM05, Theorem 5.1]) by Amano and Maruoka except for change of parameters. Assume to the contrary that there is a circuit C with at most t negations computing **CLIQUE**$(m, m^{\frac{1}{6\alpha}})$ with size M, $M < 2^{m^{\frac{1}{81\alpha}}}$. By Lemma 3.6.3 there are $\alpha \triangleq 2^{t+1} - 1$ functions $f_1, \ldots, f_\alpha$ of size at most M (as they are obtained by restrictions of the circuit C) such that $\cup_{i=1}^{\alpha} G(f_i) \supseteq G(f)$. Let $s = m^{\frac{1}{6\alpha}}$ and let $l_0, l_1, \ldots, l_\alpha$ be a monotonically increasing sequence of integers such that $l_0 = s, l_\alpha = m$ and $l_i = m^{1/10+(i-1)/(3\alpha)}$. Note that $s^{1/3} l_i \leq l_{i+1}$ as $s^{1/3} l_i = m^{1/(18\alpha)+1/10+(i-1)/(3\alpha)} < m = ^{1/10+(i)/(3\alpha)} = l_{i+1}$. Also $\left[l_0 = s = m^{\frac{1}{6\alpha}}\right] < \left[l_1 = m^{1/10}\right]$ as $\alpha = 2^{t+1} - 1 \geq 2^2 - 1$, $l_{\alpha-1} < m^{1/10+1/3} < m$. Thus, $l_0 < l_1 < \cdots < l_i < l_{i+1} < \cdots < l_\alpha$. The definition of "bad" graphs and "good" graphs at layer l_i remains the same as in [AM05]. Note that [AM05, Corollary 5.2] is true for our choice of parameters as $s^{1/3} l_{i-1} \leq l_i$. Equations 5.1 to 5.3 of [AM05] are valid in our case also as these equations does not depend on the value of the parameters. The definition of a dense set remains the same, and $h \geq \frac{1}{\alpha} \geq \frac{1}{m}$ (as $m \geq \log m \geq \alpha$) is such that $(h/2)2^{s^{1/3}/4} \geq \frac{1}{m} 2^{m^{\frac{1}{18\alpha}}/4}$ is strictly greater than $M = 2^{m^{\frac{1}{81\alpha}}}$. Hence Equation 5.4 of [AM05] is also true in our setting. Claim 5.3 of [AM05] is independent of choice of parameters, hence is true in our setting also.

Claim 3.6.6. *[AM05, Claim 5.3]*

Suppose $c_1 > 1$ and $c_2 > 1$. Put $c_3 = \alpha$. Let $f_1, \ldots, f_{c_3}$ be the monotone functions such that $\cup_{i=1}^{c_3} G(f_i) \supseteq G(\mathbf{CLIQUE}(m,s))$ and $size_{mon}(f_i) \leq M$ for any $1 \leq i \leq c_3$. Suppose that for distinct indices $i_1, \ldots, i_k \in [c_3]$,

$$\Pr_{L_k \in \mathcal{L}_k}\left[\Pr_{u \in O_{L_k}}[f_{i_1}(u) = \cdots = f_{i_k}(u) = 1] \geq \frac{1}{c_1}\right] \geq \frac{1}{c_2}$$

holds. If $c_1 c_2 c_3 \leq s_1^{1/3}/8$, then there exists $i_{k+1} \in [c_3] \setminus \{i_1, \ldots, i_k\}$ such that

$$\Pr_{L_{k+1} \in \mathcal{L}_{k+1}}\left[\Pr_{u \in O_{L_{k+1}}}[f_{i_1}(u) = \cdots = f_{i_k}(u) = 1] \geq \frac{1}{4c_1c_2c_3}\right] \geq \frac{1}{2c_1c_2}$$

Now for any $k \in [\alpha]$ there are k distinct indices $i_1, \ldots, i_k \in [\alpha]$ such that

$$\Pr_{L_k \in \mathcal{L}_k}\left[\Pr_{u \in O_{L_k}}[f_{i_1}(u) = \cdots = f_{i_k}(u) = 1] \geq \frac{1}{2^{k^2(t+2)}}\right] \quad \geq \quad \frac{1}{2^{k(t+2)}} \tag{3.2}$$

The proof is by induction on k. Base case is when $k = 1$ and follows from Equation 5.4 of [AM05] which is established to be true in our setting also. Suppose the claim holds for $k \leq l$ and let $k = l + 1$. From the induction hypothesis we get that

$$\Pr_{L_l \in \mathcal{L}_l}\left[\Pr_{u \in O_{L_l}}[f_{i_1}(u) = \cdots = f_{i_l}(u) = 1] \geq \frac{1}{2^{l^2(t+2)}}\right] \quad \geq \quad \frac{1}{2^{l(t+2)}} \tag{3.3}$$

Like in [AM05] put $c_1 = 2^{l^2(t+2)}$, $c_2 = 2^{l(t+2)}$ and $c_3 = \alpha$. Note that the bounds $4c_1c_2c_3 \leq 2^{(l+1)^2(t+2)}$, $2c_1c_1 \leq 2^{(l+1)(t+2)}$ and $c_1c_2c_3 \leq 2^{2^{3t}}/8$ are valid in our setting also as they do not depend on values of these parameters. Since $t \leq 1/6 \log\log m$, $2^{3t} \leq (\log m)^{1/3}$ and $2^{2^{3t}} \leq 2^{(\log m)^{1/3}}$ whereas $s^{1/3}$ is $m^{\frac{1}{18\alpha}} \geq 2^{(\log m)(\frac{1}{18(\log m)^{1/6}})} = 2^{(\log m)^{5/6}/18} > 2^{(\log m)^{1/3}}$. Hence $s^{1/3}/8 \geq 2^{2^{3t}}/8$. . Thus, Claim 3.6.6 applies giving us

$$\Pr_{L_{l+1}\in\mathcal{L}_{l+1}}\left[\Pr_{u\in O_{L_{l+1}}}[f_{i_1}(u)=\cdots=f_{i_{l+1}}(u)=1]\geq\frac{1}{2^{(l+1)^2(t+2)}}\right]\ \geq\ \frac{1}{2^{(l+1)(t+2)}}\qquad(3.4)$$

The proof of the main theorem is completed by noting that $\mathcal{L}_\alpha = \{V\}$ and setting k in Equation (3.2) to α gives $Pr_{u\in O_V}[\forall i \in [\alpha], f_i(u) = 1] > 0$. Thus, there exists a bad graph u belonging to $\mathbf{CLIQUE}(m,s)^{-1}(0)$ on which all of $f_1,\ldots,f_\alpha$ outputs 1, and hence (u,u^+), where $u^+ \in \mathbf{CLIQUE}(m,s)^{-1}(1)$ is a graph obtained from u by adding an edge, which is in $G(f)$ and not covered by any of the $G(f_i)$'s. A contradiction. Hence the proof. □ □

Since for a bounded fan-in circuit size lower bound of $2^{m^{\frac{1}{81\alpha}}}$ implies a depth lower bound of $m^{\frac{1}{81\alpha}}$ we get the result.

3.7 Structural restrictions on orientation

In this section we study structural restrictions on the orientation and prove stronger lower bounds.

3.7.1 Restricting the vertex set indexed by the orientation

We first consider restrictions on the set of vertices[4] indexed by the orientation - in order to prove Theorem 3.1.6. As in the other case, we argue the following lemma, which establishes the trade-off result. By using the lower bound for $\mathbf{KW}^+$ games for **CLIQUE** function, the theorem follows.

[4]Notice that the input variables to the **CLIQUE** function represents the edges. This makes the results of this section incomparable with the depth lower bounds of [RW89].

Lemma 3.7.1. If C is a circuit of depth d computing **CLIQUE**, with each gate computing a function whose orientation is such that the number of vertices of the input graph indexed by the orientation β is at most $\frac{w}{\log n}$, then d is $\Omega\left(\frac{n}{4w+1}\right)$.

Proof. It is enough to solve the $\mathbf{KW}^{+}(f)$ game on the min-term, max-term pairs which in case of **CLIQUE**(n,k) is a k-clique and a complete $k-1$-partite graph. Since we are proving a lower bound on a smaller pair of inputs, the lower bound also holds true for KWP(**CLIQUE**). We play the same game as in the proof of Theorem 3.1.2, but instead of sending edges we send vertices included in the edge set indexed by β with some additional information. If it is Alice's turn, then x'_β defines an edge sub-graph of her clique. Both Alice and Bob know β and hence know which vertices are spanned by edges $e_{u,v}$ such that $\beta_{e(u,v)} = 1$. So Alice can send a bit vector of length at most w (in the case of Alice we can handle weight of orientation up to w), indicating which of these vertices are part of her clique. This information is enough for Bob to deduce whether any $e_{u,v}$ indexed by β is present in Alice's graph or not. Since Bob makes sure that $x'_\beta = y'_\beta$ by modifying his input, and Alice keeps her input unchanged, Alice knows what modifications Bob has done to his graph.

Similarly on Bob's turn, he sends the vertices in the partition induced by y_β and the partition number each vertex belongs to (hence the $\log n$ overhead for Bob) to Alice. With this information Alice can deduce whether any $e_{u,v} \in \beta$ is present in Bob's graph or not. Inductively they maintain that they know of the changes made to other parties input in each round. Hence the game proceeds as earlier. This completes the proof of the theorem. □

3.8 Generalizing DeMorgan circuits, uniform orientation

A circuit is said to be of *uniform orientation* if there exists a single orientation vector $\beta \in \{0,1\}^N$ such that β is an orientation vector for any function which is computed by a gate of the circuit.

Note that any De-Morgan circuit is also a uniform orientation circuit with the $\beta \in \{0,1\}^n$ $\beta_i = 1$ if and only if $\bar{x_i}$ appears as a literal in the De-Morgan circuit. Hence Uniform orientation circuits generalize De-Morgan circuits.

Proposition 3.8.1. *For any function f and an orientation β of f, there is a Boolean circuit computing it with uniform orientation where the unique orientation vector associated with the circuit is β.*

Proof. Consider a function f that has orientation β with the associated monotone function being h. Let C_h be a monotone circuit computing h. Replace all the inputs $x_i \oplus \beta_i$ where $\beta_i = 1$ with $\bar{x_i}$ and all the inputs $x_i \oplus \beta_i$ where $\beta_i = 0$ with x_i in C_h. Thus, we get a circuit C computing f, which is De-Morgan and has negations only on variables in the set S, the set of indices where $\beta_i = 1$. This implies that for any function f whose orientation is β, there is a circuit C of uniform orientation β. This is because, a sub-circuit rooted at any gate g of C_g is also a De-Morgan circuit and has the set of negated variables $S' \subseteq S$. Consider β_g to be the characteristic vector of S' and the monotone function h_g to be the function computed by the monotone circuit obtained from C_g by replacing all negated variables with new variables. Hence by definition β_g is a valid orientation for g, and $\beta_g \leq \beta$ by construction. By Property 3.2.2, β is also an orientation for g. □

3.9 NC lower bounds for uniform orientation with a structural restriction

We now prove Theorem 3.1.7 from the introduction which is stated again for readability.

Theorem 3.9.1. *Let C be a circuit computing the* **CLIQUE** *function with uniform orientation $\beta \in \{0,1\}^N$ such that there is subset of vertices U and $\epsilon > 0$ such that $|U| \geq \log^{k+\epsilon} n$ for which $\beta_e = 0$ for all edges e within U, then C must have depth $\omega(\log^k n)$.*

Proof. We prove by contradiction. Suppose there is a circuit C of depth $c \log^k n$. In the argument below we assume $c = 1$ for simplicity. Without loss of generality, we assume that $|U| = \log^{k+\epsilon} n$. Fix inputs to circuit C in the following way:

- Choose an arbitrary $K_{\frac{n}{2}-\frac{|U|}{2}}$ comprising of vertices from $[n] \setminus U$ and set those edges to 1.
- For every edge in $\binom{[n]\setminus U}{2}$ which is not in the clique chosen earlier, set to 0.
- For every edge between $[n] \setminus U$ and U set it to 1.

Since every edge $e(x, y)$ which has $\beta_e = 1$ has at least one of the end points in $[n] \setminus U$, by the above setting, all those edges are turned to constants. Thus, we obtain a monotone circuit C'' computing $\mathbf{CLIQUE}(|U|, \frac{|U|}{2})$ of depth at most $(\log n)^k$. In terms of the new input, $(\log n)^k = ((\log n)^{k+\epsilon})^{\frac{k}{k+\epsilon}} = (|U|)^{\frac{k}{k+\epsilon}}$, this contradicts the Raz-Wigderson [RW92] lower bound of $\Omega(|U|)$, as $\frac{k}{k+\epsilon} < 1$ for $\epsilon > 0$. □

Note that for the Clique function, with the above corollary we can handle up to weight $\frac{n^2}{(\log n)^{2+2\epsilon}}$ if the vertices spanned by β is up to $\frac{n}{(\log n)^{1+\epsilon}}$ and still get a lower bound of $(\log n)^{1+\epsilon}$. This places us a little bit closer to the goal of handling β of weight n^2, from handling just $(\log n)^{1+\epsilon}$.

3.10 Achieving structural restrictions on orientation ϵ far from NC lower bounds

Any function has a circuit with a uniform orientation $\beta = 1^N$ ($|\beta| = N$). We show that the weight of the orientation can be reduced at the expense of depth, when the circuit is computing the **CLIQUE** function.

We now prove Theorem 3.1.8 from the introduction which is stated again for readability.

Theorem 3.10.1. *If there is a circuit C computing* **CLIQUE** *with depth d then for any set of $c \log n$ vertices U, there is an equivalent circuit C' of depth $d + c \log n$ with orientation β such that none of the edges $e(u, v)$, $u, v \in U$ has $\beta_{e(u,v)} = 1$.*

Proof. The proof idea is to devise a **KW** protocol based on circuit C such that for $e(u, v)$ where $u, v \in U$ the protocol is guaranteed to output in the monotone way, i.e., $x_{e(u,v)} = 1$ and $y_{e(u,v)} = 0$. The modified protocol is as follows:

- Alice chooses an arbitrary clique $K_{\frac{n}{2}} \in G_x$ (which she is guaranteed to find as $x \in f^{-1}(1)$). She then obtains x' by deleting edges $e(x, y)$ from $\binom{U}{2}$ which are outside the chosen clique $K_{\frac{n}{2}}$. Note that since $K_{\frac{n}{2}} \in G_{x'}$, $f(x') = 1$.
- Alice then sends the characteristic vector of vertices in $K_{\frac{n}{2}} \cap U$ which is of length at most $c \log n$ to Bob.
- Bob then obtains y' from y by removing edges in $\binom{U}{2}$ which are outside the clique formed by $K_{\frac{n}{2}} \cap \binom{U}{2}$. By monotonicity of **CLIQUE** $f(y') = 0$.
- If there is an edge $e(u, v) \in K_{\frac{n}{2}} \cap \binom{U}{2}$ which is missing from y' Bob outputs the index $e(u, v)$. Otherwise they run the standard Karchmer–Wigderson game on x', y' using the circuit C to obtain an $e(x, y)$ such that $e(x, y)$ is exclusive to either $G_{x'}$ or $G_{y'}$.

The cost of the above protocol is $d + c \log n$. For any $e(u, v) \in E(G) \setminus \binom{U}{2}$, $x'_{e(u,v)} = x_{e(u,v)}$ and $y'_{e(u,v)} = y_{e(u,v)}$. The protocol never answers non-monotonically($x'_{e(u,v)} =$

$0, y'_{e(u,v)} = 1$) for an edge $e(u,v)$ with $u, v \in U$. This is because our protocol ensures that for any $e \in \binom{U}{2}$, $x'_e \geq y'_e$. By the connection between $\mathbf{KW}(f)$ and circuit depth, we get a circuit having the desired properties. □

Thus, we get the following corollary.

Corollary 3.10.2. *If there is a circuit $C \in \mathsf{NC}^k$ computing* **CLIQUE**, *then there is a circuit $C' \in \mathsf{NC}^k$ of uniform orientation β computing* **CLIQUE** *such that there are $(c \log n)^k$ vertices V' with none of the edges $e(u,v)$ having $\beta_{e(u,v)} = 1$.*

Proof. It follows by setting $d = O((\log n)^k)$ and modifying the protocol to work over a V' of size $(c \log n)^k$. The analysis and proof of correctness of the protocol remains the same, but the communication cost becomes $O((\log n)^k) + (c \log n)^k = O((\log n)^k)$. □

In other words, if we improve Theorem 3.1.7 to the case when the orientation "avoids" a set of $\log n$ vertices (instead of $(\log n)^{(1+\epsilon)}$ as done), it will imply $\mathsf{NC}^1 \neq \mathsf{NP}$.

3.11 Discussion - "natural"-ness of orientation based depth lower bounds

We end this chapter by discussing how "natural" our proof techniques are. Karchmer Wigderson relations connect circuit depth to communication complexity of an associated relation and it is not known if this approach is "natural". The main reason is that Karchmer Wigderson relations are merely a bridge to communication complexity and there are many measures which can prove communication lower

bounds. Though we know some of them are ”natural” there are other measures which are believed not to be. Moreover, though in the case of two party communication complexity measures like Rank, Discrepancy etc., satisfy properties of a natural proof, they are known to be natural only for a function. But Karchmer Wigderson relations is inherently a relation and it is associated with not one but many communication matrices. Hence even these measures which are known to be natural for functions are not known to be natural properties for relations like Karchmer Wigderson relations. This is because the properties like Rank, Discrepancy etc are known to be constructive for functions but we do not know if they are “constructive” for relations.

CHAPTER 4

Learning sparsely oriented circuits

In this chapter we further explore the measure of non monotonicity we introduced, orientation, by studying the associated learning problem. One of the central themes of learning theory is to learn Boolean functions or, as it is known in the learning community, to learn concepts. The main idea is that there is an unknown function called the concept, say $f : \{0,1\}^n \to \{0,1\}$ that we would like to "learn", using minimal number of queries to f of the which are evaluations of f at various inputs called training data, up to an error parameter ϵ called "accuracy". Blais et. al [BCO+15a] came up with a learning algorithm under the uniform distribution membership query model for learning n variable functions computed by Boolean circuits having at most t negations to error ϵ in time $n^{O(2^t\sqrt{n}/\epsilon)}$. Using our characterization of minimal orientation of a function, and the learning algorithm of Blais et. al ([BCO+15a]) we come up with a learning algorithm under the uniform distribution membership query model for learning n variable functions computed by Boolean circuits whose weight of orientation is at most w to error ϵ in time $n^{O(w\sqrt{n}/\epsilon)}$.

Blais et. al ([BCO+15a]) also proved a matching learning lower bound by establishing for any $k : \mathbb{N} \to \mathbb{N}$ the existence of a family $\mathcal{H}^k$ of balanced[1] $k(n)$ alternating[2] functions such that, for any sufficiently large n, $\epsilon > 0$, $2 \leq k \leq n^{1/4}$ and $k^{7/3}/n^{1/6} \leq \epsilon \leq 1/2 - c$, learning $\mathcal{H}^k$ to accuracy $1 - \epsilon$ requires $2^{\Omega(k\sqrt{n}/\epsilon)}$. Using our characterization of orientation, we show that this family of functions $\mathcal{H}^k$ has

[1] A Boolean function f is said to be balanced if $|f^{-1}(0)| = |f^{-1}(1)|$

[2] on any chain of the hypercube function changes from 1 to 0 on at most $k(n)$ edges

minimal weight of orientation equal to the number of inputs, the maximum possible. Thus, we show that their lowerbound strategy does not give lower bounds for learning sparsely oriented circuits.

Based on the techniques employed in the lower bound of [BCO+15a] i.e., hardness amplification of learning under composition of functions [FLS11], we come up with an information theoretic noise model similar to that of [O'D04]. We suggest expected bias under this noise model as the right parameter for studying the hardness amplification of composition of functions. We show a partial result suggesting that expected bias under the newly defined noise model is a right parameter for certain families of top functions, like REC-3-MAJ. Based on this intuition we also state a conjecture on hardness amplification of learning a specific composition of functions tailored for sparse orientation.

Though we are unable to obtain a learning lower bound for limited orientation functions, we conclude the chapter with important open problems which when solved, assuming the conjecture, would give such a lower bound but are also of independent interest in Fourier analysis.

The results that appear in this chapter are from our work that appear in [Kor17].

4.1 Limited non-monotonicity and learning

Monotone functions are central to learning theory. One of the early breakthrough algorithms for learning a class of functions, was the uniform distribution learning algorithm for monotone functions ([BT96]) devised by Bshouty and Tamon. Thus, a natural step is to study the learning problem for non-monotone Boolean functions. Blais et. al ([BCO+15a]) studied learning functions computed by circuits with

at most t negations, under the uniform distribution model. They proved near matching upper and lower bounds for this learning problem.

Their uniform distribution learning algorithm is based on an alternate characterization of inversion complexity of Boolean functions. They proved that if a function can be computed by at most t negations, then it can be written as parity or complement of parity of t threshold functions. They use this characterization to bound a Fourier analytic parameter of the function called influence as function of t. They combine the bound on influence along with the algorithm for learning monotone functions by Bshouty and Tamon [BT96], also based on influence of the function, to get a uniform distribution learning algorithm.

To prove the matching lower bound, they use the idea of hardness amplification of learning under the composition of functions. The high level overview of the argument is to first prove strong lower bounds for learning monotone functions to "high" accuracy. Using hardness amplification for learning from [FLS11], the authors lift it to a learning lower bound for learning monotone functions up to "moderate" accuracy. The hardness amplification for learning theorem from [FLS11] is based on O'Donnell's work on on hardness amplification within NP [O'D04]. The basic idea to lift the hardness of a given function h using a function g, which is preferably monotone or even of limited inversion complexity, is by feeding multiple independent copies of h to g. The setting is similar to that of the XOR-lemma, but as $\oplus$ has the highest inversion complexity possible over all n-bit functions it is not particularly suited for the purpose of lifting monotone results to monotone, or to limited inversion complexity. The main intuition is that to lift the hardness under composition, we can use a function g which has high "noise sensitivity". A function is noise sensitive if its value changes with high probability

when its inputs are perturbed with small "random noise".

To lift the result of learning monotone functions to "moderate accuracy" to that about learning limited negation functions to "moderate accuracy", Blais et. al ([BCO+15a]) use a function which acts like the $\oplus$ function in the middle layers of the hyper cube and is thus very sensitive to noise. This function also has the desired inversion complexity and hence the hardness amplification lemma ([FLS11, Theorem 12]) gives the matching lower bound for learning limited negation functions.

In this chapter, we study the learning problem for sparsely oriented functions. The motivation is that orientation is another measure of non-monotonicity of Boolean functions. And like inversion complexity of functions, we have proved in Chapter 3 that it is a robust measure. Like Markov's characterization of inversion complexity in terms of alternation number of the hypercube, we characterized orientation in terms of the axis of non-monotone edges in the hypercube.

Because of the characterization of orientation, we can show that a weight orientation at most w function is computed by a circuit of at most $\log w$ negations. And hence the learning upper bound for functions computed by $\log w$ negations from [BCO+15a] gives a learning upper bound for w-orientation functions.

Next we turn to the lower bound problem. We first show why the lower bound approach of [BCO+15a] et. al doesn't work for us. The main hurdle is in the last stage of their argument where multiple copies of a monotone function h are fed as inputs of a function g which is of low inversion complexity. Such a composition doesn't increase the inversion complexity of the combined function, but the orientation of the combined function can increase arbitrarily.

An intuitive strategy to get around the difficulty is to compose g with two kinds of functions, one which is monotone, and another which is of limited orientation

w. We turn to the original hardness amplification setting of O'Donnell [O'D04] and come up with an information theoretic noise model for composing functions tailored for orientation. We suggest Expected Bias under this noise model as the right parameter for hardness amplification. We also establish a theorem which furthers the intuition as to why expected bias is the right parameter for our noise model, for certain classes of functions. But in the "noise regime" we propose, analyzing expected bias becomes fairly complicated as one function is harder than the other, resulting in the coordinates noisier than the others. We conclude the chapter by stating a conjecture which can help establish learning lower bounds for sparsely oriented circuits.

The rest of the chapter is organized as follows. In Section 4.2, we give the details of the learning algorithm for sparsely oriented functions. In the next section, Section 4.3 we outline a specific composition of functions for hardness amplification tailored for orientation and an information theoretic noise model for this composition.

4.2 A learning algorithm

In this section we show how to use the properties of orientation along with the limited negation learning algorithm to obtain an algorithm for learning functions of sparse orientation.

Recall that we say that a circuit C is weight w oriented if every internal gate of C computes a function which has an orientation β with $|\beta| \leq w$. The weight restriction on a circuit thus defined is a semantic restriction as we are only limiting the weight of orientation of the functions computed at sub-circuits of C. But we

do not place any restriction on how (especially in terms of actual negation gates) the functions at sub-circuits are computed in C.

We use the property of orientation that the orientation vector corresponding to the minimum weight of a function is unique, proved in Chapter 3 (Property 3.2.4), to get a learning algorithm from the learning algorithm for learning circuits with limited negation gates. Let C_w^n denote the family of circuits with n inputs and of weight of orientation at most w. For any Boolean function f computed by such a circuit the weight of orientation at most w. This is because the root gate of such a circuits computes f and has weight of orientation like any other internal gate of the circuit at most w.

We now show that any function of orientation at most w can be computed with a circuit having at most $\log w$ negations. As a first step we show that there is a De-Morgan circuit computing f with at most w negated literals. Then we use a result of Fischer [Fis75] to get a circuit which computes all these w negated literals using at most $\log w$ negations.

Lemma 4.2.1. For any function f of weight of orientation at most w, there exists a circuits which computes f with at most $\log w$ negations

Proof. Since the minimal weight of orientation of f is w, by Property 3.2.4, there exists a monotone function h and a unique orientation vector β having at most w ones such that, $\forall x \in \{0,1\}^n, f(x) = h(x, (x \oplus \beta))$. Consider any monotone circuit computing h. To correctly compute f, h additionally needs only w negated variables, say $x_{i_1}, \ldots, x_{i_w}$. By a result of Fischer [Fis75] these negated variables can be computed by a circuit having at most $\log w$ negations. Hence it follows that f is computed by a circuit of at most $\log w$ negations. □

This combined with the learning algorithm of Blais et. al [BCO+15b] stated below gives a straight forward learning algorithm for limited orientation circuits.

Theorem 4.2.2 (Theorem 1.4, [BCO+15b])**.** *There is a uniform-distribution learning algorithm which learns any unknown Boolean function f which is computed by a circuit of at most t negations, from random examples to error ϵ in time $n^{O(2^t \sqrt{n}/\epsilon)}$.*

We now prove the learning upper bound for sparsely oriented circuits.

Theorem 4.2.3. *There is a uniform-distribution learning algorithm which learns any unknown Boolean function f which is computed by a circuit of at most w weight of orientation, from random examples to error ϵ in time $n^{O(w \sqrt{n}/\epsilon)}$*

Proof. By Lemma 4.2.1, f is computed by a circuit having at most $\log w$ negations. Hence the Theorem 4.2.2 of Blais et. al [BCO+15a] with $t = \log w$ implies the theorem.

□

4.3 Towards learning lower bounds : noise models

4.3.1 Overview of noise model used by Blais et. al

In this section we give a high-level overview of the hardness amplification based learning lower bound in [BCO+15b]. They first prove a lower bound for learning monotone circuits to high accuracy. Since the algorithm has to learn to really high accuracy (read constant error), the following simple claim is true.

Claim 4.3.1. *Claim 3.13 of [BCO+15a] There exists a class of balanced monotone Boolean functions $\mathcal{G} = \{G_m\}_{m \in \mathbb{N}}$ and a universal constant C such that, for any constants $0 < \alpha \leq$*

1/10, *learning* G_m *to error* $0 < \epsilon \leq \alpha/\sqrt{m}$ *requires at least* $2^{C \times m}$ *queries.*

The proof is an adversarial argument where they consider a slice function as the candidate function for the lower bound. A *slice monotone function* on n-bits (assume that n is even) is a function which is guaranteed to be 0 on inputs which have strictly less than $n/2$ 1's and 1 on inputs which have strictly more than $n/2$ 1's in them. The function can take any value on the middle layer of the hypercube, i.e., inputs x which have exactly $n/2$ 1's in them. The proof relies on the fact that, the middle layers contains almost a constant fraction of the total number of points in the hypercube. Hence any learning algorithm which queries only 2^{Cm} points for an appropriate value of C, misses a constant fraction of points in the middle layer. Hence the adversary can set these points arbitrarily to increase the error.

Now the authors use composition with a function which is very stable under noise to get a function which is hard to learn even up to "moderate" accuracy.

The hardness-amplification scenario used here is similar to the one used by O'Donnell [O'D04] for hardness amplification within NP. We start with the "mildly" hard function we obtained earlier, say $h : \{0,1\}^m \to \{0,1\}$, and compose it with a function, say $g : \{0,1\}^r \to \{0,1\}$, which is very sensitive to noise. The composition is the standard composition of two Boolean functions denoted by $g \otimes h : \{0,1\}^{rm} \to \{0,1\}$, defined as $g \otimes h(x_1, \ldots, x_r) = g(h(x_{11}, \ldots, x_{1m}), \ldots, h(x_{r1}, \ldots, x_{rm}))$. Noise sensitivity of g is measured using $\text{ExpBias}_\delta(g)$. Before we define ExpBias, we need to define the bias(f) of a Boolean function. The bias(f) as the name suggests is the absolute difference between number of zeros and ones of f, more formally $\text{bias}(f) = \| f^{-1}(0) | - | f^{-1}(1) \|$. Thus, $\text{ExpBias}_\delta(g)$ is defined as the expectation of the bias of g under random restrictions where each input of g is left unrestricted with probability δ, set to 0 with probability $(1-\delta)/2$, and to 1 with

probability $(1-\delta)/2$. More formally, $\text{ExpBias}_\delta(g) = \mathsf{E}_{\rho \in P^r_\delta}\left[\text{bias}(g_\rho)\right]$ where $\rho \in P^m_\delta$ is a random restriction which leaves each of the r inputs of g unfixed with probability δ and sets them to 1 or 0 with equal probability otherwise.

The reason $\text{ExpBias}_\delta(g)$ is the right parameter for studying hardness amplification of composition of functions is the following information theoretic scenario modeling average case hardness of functions. The hardness of computing or learning each $h(x_i)$ is modeled by setting the input to g, say z_i to 0 with probability $(1-\delta)/2$ and to 1 with probability $(1-\delta)/2$. And the coordinate z_i is un-restricted with probability δ. This models the scenario that there is a δ fraction of inputs on which h is balanced and where h is impossible to compute, and hence $z_i = h(x_i)$ looks like a random bit to the learning algorithm or the circuit trying to compute $g \otimes h$. On the remaining $1-\delta$ fraction, we even assume in our model the circuit can correctly compute/learn the function. The assumption of hard core set on which the function h has equal number of zeros and ones and is almost random to the circuit/learning algorithm is based on the Hardcore set lemma of Impagliazzo [Imp95] adapted by [O'D04] to suit the context of hardness amplification within NP. Thus, in this information theoretic scenario of hardness, composing a $1-\delta$ hard function h with a function g is reduced to correctly computing g on inputs z_i where each z_i is set to 0 with probability $(1-\delta)/2$, to 1 with probability $(1-\delta)/2$ and is un-restricted with probability δ. Thus, we are assuming even that the circuit knows which bits z_i are random and which are set. In this scenario, it seems like the best strategy for the circuit/learning algorithm to compute g on $z_1, \ldots, z_r$, is to output the $\text{bias}_\rho(g)$. [O'D04] proves this formally by showing that any probabilistic procedure which tries to guess the value of g, cannot have an advantage ϵ more than $\text{bias}(g)$ unless it can distinguish between two end points of a hyper-edge[3] with non-negligible

[3] $x, y \in \{0,1\}^n$ such that x and y differ only in one bit

probability.

Then [O'D04] uses this idea to prove that the expected bias of g essentially characterizes the hardness of computing the composed function $g \otimes h$. Feldman et. al [FLS11] generalize this idea to the setting of hardness amplification for learning by proving the following theorem.

Theorem 4.3.2 (Theorem 12 of [FLS11]). *Fix $g : \{0,1\}^r \to \{0,1\}$, and let $\mathcal{F}$ be a class of m-variable Boolean functions such that for every $f \in \mathcal{F}$, bias$f \leq \frac{1}{2} + \frac{\epsilon}{8r}$. Let A be a uniform distribution membership query algorithm that learns $g \otimes \mathcal{F}$ to accuracy ExpBias$_\gamma(g) + \epsilon$ using $T(m, r, 1/\epsilon, 1/\gamma)$ queries. Then there exists a uniform-distribution membership query algorithm B, that learns $\mathcal{F}$ to accuracy $1 - \gamma$ using $O(T(\mathsf{poly}(m, r, 1/\epsilon, 1/\gamma))$ membership queries.*

Another important idea that both [FLS11] and [BCO+15a] derive from [O'D04] is that, though estimating expected bias of a Boolean function is hard. But another well known Fourier analytic parameter of Boolean functions called noise stability is a good approximation for expected bias. The noise stability, denoted by NoiseStab$_\delta(g)$, is defined to be the probability that the value of g doesn't change for a random input when it is perturbed by a noise δ. Given an $x \in \{0,1\}^r$, the noise δ applied to x is a random variable denoted by $N_\delta(x)$ obtained by flipping each bit of x with probability δ. Thus, the formal definition of NoiseStab$_\delta(g)$ = Pr$[g(x) = g(N_\delta(x))]$ where the probability is over a uniformly and randomly chosen x from $\{0,1\}^r$ and the noise. Note that all quantities discussed so far like bias, expected-bias, noise stability are all in the range $[1/2, 1]$. For any quantity Q in the range $[1/2, 1]$, the corresponding quantity Q^* defined to be $Q^* = 2(Q - 1/2)$ takes it to the range $[0, 1]$. This transformation is helpful in capturing the relationship between noise stability and expected bias. [O'D04]

proved that $\mathrm{NoiseStab}_{\delta}(g)^{*} \leq \mathrm{ExpBias}_{2\delta}(g)^{*} \leq \sqrt{\mathrm{NoiseStab}_{\delta}(g)^{*}}$.

The first step in lifting the hardness of learning monotone functions to high accuracy to that of learning monotone functions to moderate accuracy, is the application of the above theorem with a monotone function g which is balanced and has very small noise stability. This implies that the expected bias of the function is very low, and hence learning this function even to moderate error is impossible. The function g used in the composition $g \otimes h$ is the monotone function obtained by Mossel and O'Donnell in [MO03]. After doing this the authors obtain a family of functions which are balanced, monotone and hard to learn even to moderate accuracy.

The second step is to lift this hardness to a function which has limited inversion complexity, i.e., one which is computed by a circuit with few negations. Blais et. al [BCO+15b] uses a top function g which is the parity function restricted to k-layers around the middle layer of the Boolean hypercube of dimension k^2. Formally,

Definition 4.3.3. For any odd $r \geq k \geq 1$, let $\mathrm{PAR}'_{k,r}$ be the symmetric Boolean function on r inputs defined as follows: for all $x \in \{0,1\}^r$,

$$\mathrm{PAR}'_{k,r}(x) = \begin{cases} 0 & |x| > \frac{r+k}{2} \\ 0 & |x| < \frac{r-k}{2} \\ \mathrm{PAR}_r(x) & \text{Otherwise} \end{cases}$$

By showing PAR'_{k,k^2} has enough correlation to parity on k bits, Blais et. al ([BCO+15a]) establish that this function has low noise stability and thus low expected bias. Hence one more application of Theorem 4.3.2 gives them the desired result, as PAR'_{k,k^2} can be computed with $\log k$ negations. The final function which

is a composition of a monotone function with PAR'_{k,k^2} can also be computed with $\log k$ negations.

4.3.2 A candidate noise model for orientation

Unlike the number of negation gates in a circuit, which is a structural restriction, orientation is a semantic restriction. Thus, the non-monotonicity of the top composition function can affect the non-monotonicity of the bottom part after composition. We note that though PAR'_{k,k^2} requires only $\log k$ negations (as it is k-alternating), it has maximum possible orientation. This follows from the Proposition 3.2.3 of Chapter 3 and the fact that for any $i \in [k^2]$ we can find an $(x, x + e_i)$ such that PAR'_{k,k^2} changes anti-monotonically. By definition of PAR'_{k,k^2} and the fact that it is symmetric, either $\text{PAR}'_{k,k^2}(0^{k^2/2}1^{k^2/2})$ or $\text{PAR}'_{k,k^2}(0^{k^2/2-1}1^{k^2/2+1})$ must be 1 and the entire layer above it will be all 0's. Hence for any $i \in [k^2]$ we can find an $(x, x+e_i)$ where change happens anti-monotonically. This is a problem as we do not get any guarantees on the orientation of the final function after this step, other than the naive bound that it might be the maximum possible. Hence the approach of Blais et. al [BCO+15a] does not give a lower bound for learning Boolean functions whose orientation is limited.

But we note that this approach gives a learning lower bound for functions whose orientation is the maximum possible.

Since any function on n variables has weight of orientation at most n, we get the following easy lemma by setting k to be the maximum possible value in Corollary 3.5 from [BCO+15a].

Lemma 4.3.4. Learning the class of weight of orientation n functions to accuracy

$1-\epsilon$ in the uniform distribution membership-query model requires $2^{\Omega((n^{1/28}\sqrt{n})/\epsilon)}$ membership queries, for any $\epsilon \in [1/n^{1/12}, 1/2 - c]$

Proof. By setting $k = n^{1/28}$ in 3.5 from [BCO+15a] we get a learning lower bound $2^{\Omega((n^{1/28}\sqrt{n})/\epsilon)}$ membership queries for the family of $n^{1/28}$-alternating functions. Any function in this family has n inputs and thus can have weight of orientation at most n. Hence learning weight of orientation at most n functions is at least as hard as learning this family. □

A natural strategy to get around the problem of arbitrary orientation as a result of composing a non-monotone function with a monotone function in [BCO+15a], is to use the top function g in the second application of Theorem 4.3.2 to be monotone. And to get a final function which has limited weight of orientation, we can compose g with two functions f_1 and f_2. We choose f_1 to be the hard to learn (even up to moderate accuracy) function with maximum weight of orientation from Lemma 4.3.4 and f_2 to be the hard to learn (even up to moderate accuracy) monotone function from [BCO+15a]. This way we can ensure that the orientation of the final function is not the maximum, and is controlled by the number of f_1's we choose to feed to g. Hence we are interested in the following hardness amplification scenario (and associated learning lower bounds) :

Let $g : \{0,1\}^k \to \{0,1\}$ be a Boolean function. And let $f_1 : \{0,1\}^n \to \{0,1\}$ and $f_2 : \{0,1\}^n \to \{0,1\}$ be two Boolean functions which are $1-\delta_1$ and $1-\delta_2$ hard respectively for size s circuits. The hardness amplification considers the function $g \otimes^{N_1} (f_1, f_2) : \{0,1\}^{nk} \to \{0,1\}$ for any $N_1 \subseteq [k]$ and is defined as follows :

$$g \otimes^{N_1} (f_1, f_2)(x_1, \ldots, x_k) \triangleq g(h_1(x_1), \ldots, h_k(x_k))$$

where for any $i \in [k]$, h_i is the function f_1 if $i \in N_1$ and is the function f_2 otherwise.

The candidate noise model : In the information theoretic hardness scenario defined in [O'D04], this would correspond to an adversary trying to compute g on "corrupted" bits $z_1, \ldots, z_k$ where for any $i \in N_1$, $z_i = f_1(x_i)$ with probability $1 - 2\delta_1$ and in this case the adversary knows that the bit is correct (as in the noise model of [O'D04]). Otherwise it is a random bit which has no correlation with $f_1(x_i)$, with probability $2\delta_1$. For any $i \in [k] \setminus N_1$, $z_i = f_2(x_i)$ with probability $1 - 2\delta_2$ and in this case as before we know the bit is correct, and is a random bit which has no correlation with $f_2(x_i)$ with probability $2\delta_2$ otherwise.

It is natural to assume that the best strategy in this case, as in the noise model of [O'D04], is for the adversary to guess the value of g according to the expected bias of g under restriction ρ sampled uniformly at random from $P^k_{N_1,\delta_1,\delta_2}$ defined below.

$$\rho(i) = \begin{cases} \star & \text{with probability } 2\delta_1, i \in N_1 \\ 0 & \text{with probability } \frac{1-2\delta_1}{2}, i \in N_1 \\ 1 & \text{with probability } \frac{1-2\delta_2}{2}, i \in N_1 \\ \star & \text{with probability } 2\delta_2, i \in [k] \setminus N_1 \\ 0 & \text{with probability } \frac{1-2\delta_2}{2}, i \in [k] \setminus N_1 \\ 1 & \text{with probability } \frac{1-2\delta_2}{2}, i \in [k] \setminus N_1 \end{cases}$$

In the rest of the chapter we will denote by $\mathrm{ExpBias}_{N_1,\delta_1,\delta_2}(g) = \mathsf{E}_{\rho \in P^k_{N_1,\delta_1,\delta_2}}\left[\mathrm{bias}(g_\rho)\right]$.

Under the noise model defined above, the success probability of the best strategy is intuitively the expected bias of g. To prove that guessing according to the expected bias of g is the best strategy for an adversary, we need to show that if

there is a probabilistic procedure p guessing the value of g_ρ on a randomly chosen input of the hypercube, which has advantage more than $\text{bias}_\rho(g) + \epsilon$, then it can detect at least one coordinate $i \in [k_1]$ with advantage better than δ_1 and at least one coordinate $i \in [k_2]$ with advantage better than δ_2. We do not know how to prove this for a general g. But if the function g is symmetric, we can show that one of the key ingredients, the proof of [O'D04, see Lemma 5] can be extended to our noise model. Recall that the Lemma 5 of [O'D04] claims that :

Lemma 4.3.5. Lemma 5 [O'D04] Let $\mathfrak{B}_k$ denote the k-dimensional Hamming cube, and suppose $h : \mathfrak{B}_k \to \{0,1\}$ and $p : \mathfrak{B}_k \to [0,1]$. Further suppose that,

$$2^{-k}\left[\sum_{y \in h^{-1}(0)} p(y) + \sum_{y \in h^{-1}(1)} (1 - p(y))\right] \geq \text{bias}(h) + \varepsilon$$

Then there exists an edge (z, z') in $\mathfrak{B}_k$ such that $|p(z) - p(z')| = \Omega\left(\varepsilon / \sqrt{k}\right)$

But it could be that all such edges (z, z') come from a specific dimension i alone, which in the worst case could be fed by a not-so-hard to learn monotone function. But we can make sure that this does not happen if the function h is a symmetric function. More formally we prove the following theorem.

Theorem 4.3.6. *Let $\mathfrak{B}_k$ denote the k-dimensional Hamming cube, and suppose $h : \mathfrak{B}_k \to \{0,1\}$ is a symmetric function and $p : \mathfrak{B}_k \to [0,1]$. Further suppose that,*

$$2^{-k}\left[\sum_{y \in h^{-1}(0)} p(y) + \sum_{y \in h^{-1}(1)} (1 - p(y))\right] \geq \text{bias}(h) + \varepsilon$$

Then for any dimension $i \in [k]$ there exists a $p' : \mathfrak{B}_k \to [0,1]$ and a dimension i edge (z, z') in $\mathfrak{B}_k$ such that $|p'(z) - p'(z')| = \Omega\left(\varepsilon / \sqrt{k}\right)$

Proof. Let (z, z') be the edge in $\mathfrak{B}_k$ of say dimension j, obtained by applying

Lemma 4.3.5 to the distinguisher p. We would now like to come up with with another edge for any other dimension i. Let $\pi \in S_k$ be $\pi(i) = j, \pi(j) = i$ and for any $l \in [k], l \neq i, l \neq j$, $\pi(l) = l$. Define a new distinguisher p' such that $p'(z) = p(\pi(z))$. We now show that the distinguisher p' has at least the advantage of the distinguisher p.

$$\begin{aligned}
2^{-k}\left[\sum_{y\in h^{-1}(0)} p'(y) + \sum_{y\in h^{-1}(1)} (1-p'(y))\right] &= 2^{-k}\left[\sum_{y\in h^{-1}(0)} p(\pi(y)) + \sum_{y\in h^{-1}(1)} (1-p(\pi(y)))\right] \\
&= 2^{-k}\left[\sum_{\pi(y)\in h^{-1}(0)} p(\pi(y)) + \sum_{\pi(y)\in h^{-1}(1)} (1-p(\pi(y)))\right] \\
&\quad (\because \quad h \text{ is symmetric}, h(\pi(y)) = h(y)) \\
&\geq \text{bias}(h) + \varepsilon
\end{aligned}$$

By Lemma 4.3.5, edge (z, z') has $|p(z) - p(z')| = \Omega\left(\varepsilon/\sqrt{k}\right)$. But that means for $(\pi^{-1}(z), \pi^{-1}(z'))$, we have that $|p'(\pi^{-1}(z)) - p'(\pi^{-1}(z'))| = |p(\pi(\pi^{-1}(z))) - p(\pi(\pi^{-1}(z')))| = |p(z) - p(z')| = \Omega\left(\varepsilon/\sqrt{k}\right)$. Note that $(\pi^{-1}(z), \pi^{-1}(z'))$ is a dimension i edge. This is because (z, z') is an edge of dimension j, and π as well as π^{-1} maps j to i and vice versa. □

Symmetric functions, like MAJ, have high noise stability whereas for the hardness amplification we need top functions of low noise stability. But we can exploit the fact that the above proof works even if g was "locally symmetric" like Rec-3-Majority function. The Rec-3-Majority function, as the name suggests, groups its input bits into chunks of 3 bits and then computes the majority of each chunk and then again groups these bits into chunks of 3 bits and computes the majority of each such chunk, and so on and so forth until a single bit is obtained. More formally, REC-3-MAJ$(x_{11}, \ldots, x_{n3})$ = REC-3-MAJ$(\text{MAJ}_3(x_{11}, x_{12}, x_{13}), \ldots, \text{MAJ}_3(x_{n1}, x_{n2}, x_{n3}))$. This function is locally symmetric, i.e., if any two inputs to any of MAJ_3 at the bottom

are swapped the function value remains the same. We thus define what it means to be locally symmetric as follows:

Definition 4.3.7. A function $g : \{0,1\}^r \to \{0,1\}$ is said to be locally symmetric with respect to a subset $N_1 \subseteq [r]$, if there is $\pi \in S_r$ and indices $i_1, i_2 \in N_1$, $j_1, j_2 \in [n] \setminus N_1$ such that for all $x \in \{0,1\}^r$, $g(x) = g(\pi(x))$ and $\pi(i_1) = j_1$ and $\pi(j_2) = i_2$.

To exploit the local symmetry of REC-3-MAJ, we would feed every chunk of 3 bits at the bottom level with one input from f_1 and the other two inputs from f_2. But since we are feeding in every three inputs, two inputs from f_2, the resulting composed function can have weight orientation as high as $1/3$ of the input size. By changing Rec-3-Majority with Rec-c-Majority we can obtain lower bounds for learning functions of weight of orientation at most $1/c$ of the maximum possible. Thus, our lower bound strategy works for weight of orientation a multiplicative factor of the input size.

But there is another issue with our strategy in general. Since our strategy crucially depends on the top function being monotone, the best noise sensitivity we can obtain is the best that can be achieved by a monotone function. Mossel and O'Donnel [MO03] proved an upper bound on noise sensitivity of monotone functions by proving a lower bound on their noise stability. More formally they proved that for any function $g : \{0,1\}^r \to \{0,1\}$ which is monotone, its noise stability under a noise operator N_δ, is at least $(1-\delta)^{\sqrt{2/\pi}+o(1)\sqrt{r}}$. In our setting of noise, there are two operators, N_{δ_1} and N_{δ_2} where $\delta_1 < \delta_2$. It is easy to see that the noise stability under this operator is at least the noise stability under the standard noise operator N_{δ_2} as each coordinate is flipped with equal or higher probability than earlier. Thus, the minimum noise stability that can be achieved is at least $(1-\delta_2)^{\sqrt{2/\pi}+o(1)\sqrt{r}}$. This limits the potential lower bound that can be achieved to be $2^{\Omega(\sqrt{w}\sqrt{n}/\epsilon)}$ instead

of $2^{\Omega(w\sqrt{n}/\epsilon)}$. Hence we would like the top function to have as small a noise stability as possible while still being locally symmetric. We can achieve this by using the monotone function obtained by Mossel and O'Donnel (Theorem 3 of [MO03]) which has noise stability close to the lower bound. But this function is not locally symmetric. To circumvent this issue we can consider the function from [MO03] combined with majority of three inputs each where we would feed every chunk of 3 bits at the bottom level with one input from f_1 and other two inputs from f_2. Since the top function is a function of high noise sensitivity and majority is applied only on 3 bits and also is a function of high noise sensitivity it is safe to assume that even under the new noise model the combined top function has high noise sensitivity.

Towards a learning lower bound using the noise model : Based on the intuition from the noise model, and the Theorem 4.3.6 supporting expected bias under the noise model as the right parameter for hardness amplification, we make a conjecture about hardness amplification for learning under this noise model, along the lines of the Theorem 12 of Feldman et. al [FLS11].

Conjecture 4.3.8. *Fix $g : \{0,1\}^r \to \{0,1\}$ and subset $N_1 \subseteq [r]$ such that g is locally symmetric with respect to N_1. Let $\mathcal{F}_1, \mathcal{F}_2$ be classes of m-variable Boolean functions such that for every f from $\mathcal{F}_1$ or $\mathcal{F}_2$, $\text{bias} f \leq \frac{1}{2} + \frac{\epsilon}{8r}$. Let A be a uniform distribution membership query algorithm that learns $g^{\otimes N_1}(\mathcal{F}_1, \mathcal{F}_2)$ to accuracy $ExpBias_{\gamma(N_1,\delta_1,\delta_2)}(g) + \epsilon$ using $T(m, r, 1/\epsilon, 1/\gamma)$ queries. Then there exists a uniform-distribution membership query algorithm B_1, that learns $\mathcal{F}_1$ to accuracy $1 - \delta_1$ using $O(T.\mathsf{poly}(m, r, 1/\epsilon, 1/\gamma))$ queries. Also there exists a uniform-distribution membership query algorithm B_2, that learns $\mathcal{F}_2$ to accuracy $1 - \delta_2$ using $O(T.\mathsf{poly}(m, r, 1/\epsilon, 1/\gamma))$ queries.*

Assuming the conjecture above, all that is to be done to prove a learning lower bound for sparsely oriented circuits is to analyze the expected bias of a monotone function g under the noise model we defined earlier, $\mathrm{ExpBias}_{\gamma(N_1,\delta_1,\delta_2)}(g)$. But we do not know how to analyze the expected bias or noise stability of functions under the noise operator we defined. The noise operator we study is far more complicated than the one used by [O'D04]. The ρ we defined can be thought of as two different noise operators on two different set of inputs. It is no more clear how to analyze noise stability of functions under ρ as ρ acts as different noise on different part of inputs. We point out why the two methods of estimating noise stability employed by [O'D04] don't work for our noise regime. The way [O'D04] estimates noise stability of a function like REC-3-MAJ is using the fact that if h is a balanced Boolean function, and g is any Boolean function, then $\mathrm{NoiseStab}_{\delta}(g \otimes h) = \mathrm{NoiseStab}_{1-\mathrm{NoiseStab}_{\delta}(h)}(g)$. This fact follows straight forwardly from the definition of NoiseStab itself. But we do not know how to prove such a theorem under the noise operator we defined and it does not follow from the definition. Another approach [O'D04, Proposition 9] to bound the noise stability uses the Fourier analytic characterization that $\mathrm{NoiseStab}_{\delta}(h)^* = \sum_{S\subseteq[n]}(1-2\delta)^{|S|}\hat{h}(S)^2$. Let us denote by $N_1 \subseteq [n]$, the indices where the noise operator N_{δ_1} acts, and $N_2 = [n] \setminus N_1$ be the indices where the other noise operator, N_{δ_2} acts. The equivalent expression for noise stability under this regime becomes, $\mathrm{NoiseStab}_{N_1,\delta_1,\delta_2}(h)^* = \sum_{S\subseteq[n]}(1-2\delta_1)^{|S\cap N_1|}(1-2\delta_2)^{|S\cap N_2|}\hat{h}(S)^2$. We do not know how to analyze this expression to get an upper bound for noise stability. We leave this as an interesting open problem.

Open Problem 1. Analyze the expected bias of REC-3-MAJ : $\{0,1\}^k \to \{0,1\}$ under the noise model we defined. That is analyze,

$$\mathrm{ExpBias}_{\gamma(N_1,\delta_1,\delta_2)}(\text{REC-3-MAJ}) = \mathsf{E}_{\rho\in P^k_{N_1,\delta_1,\delta_2}}\left[\mathrm{bias}(\text{REC-3-MAJ}_\rho)\right]$$

where $\rho \in P^k_{N_1,\delta_1,\delta_2}$ is defined to be,

$$\rho(i) = \begin{cases} \star & \text{with probability } 2\delta_1, i \in N_1 \\ 0 & \text{with probability } \frac{1-2\delta_1}{2}, i \in N_1 \\ 1 & \text{with probability } \frac{1-2\delta_2}{2}, i \in N_1 \\ \star & \text{with probability } 2\delta_2, i \in [k] \setminus N_1 \\ 0 & \text{with probability } \frac{1-2\delta_2}{2}, i \in [k] \setminus N_1 \\ 1 & \text{with probability } \frac{1-2\delta_2}{2}, i \in [k] \setminus N_1 \end{cases}$$

As pointed out earlier REC-3-MAJ might not have good enough noise stability to get $2^{\Omega(\sqrt{w}\sqrt{n}/\epsilon)}$ lower bound for learning circuits of orientation at most w. But we believe that it is an easier function to analyze than the the monotone function obtained by Mossel and O'Donnel (Theorem 3 of [MO03]). This is because the function of [MO03] is based on a probabilistic construction of Talagrand [Tal96] and is a random CNF formula. But to obtain the best lower bound we would like to analyze the expected bias of the Mossel and O'Donnels function. To exploit the intuition of Theorem 4.3.6, we would like to make this function locally symmetric, as suggested earlier, without loosing much of its noise sensitivity. Let $g : \{0,1\}^r \to \{0,1\}$ be the function in Theorem 3 of [MO03]. And let $h : \{0,1\}^c \to \{0,1\}$ be the MAJ_c function. The monotone function $f_c : \{0,1\}^{rc} \to \{0,1\}$ is defined to be $f(x_{1,1},\ldots,x_{1,c},\ldots,x_{r,c}) = g(h(x_{1,1},\ldots,x_{1,c}),\ldots,h(x_{r,1},\ldots,x_{r,c}))$. The function f is locally symmetric among the coordinates $(x_{i,1},\ldots,x_{i,c})$ for any $i \in [r]$. Thus, we would like to analyze the expected bias of this function under an $N_1 \subseteq [rc]$, of coordinates where the first noise operator N_{δ_1} operates, which exploits local symmetry of f. We provide a non-trivial, yet simple case which would help in obtaining a learning lower bound for circuits of weight of orientation at most 1/3rd of the

maximum.

Open Problem 2. For $c = 3$, $N_1 = \{(i,1),(i,2)\}_{i\in[r]}$, analyze the expected bias of $f_3 : \{0,1\}^{3r} \to \{0,1\}$ under the noise model we defined. That is analyze,

$$\mathrm{ExpBias}_{\gamma(N_1,\delta_1,\delta_2)}(f_3) = \mathsf{E}_{\rho\in P^k_{N_1,\delta_1,\delta_2}}\left[\mathrm{bias}(f_{3\rho})\right]$$

To summarize, our strategy can be executed to get a learning lower bound for Boolean functions whose weight of orientation is limited, assuming Conjecture 4.3.8, provided one can analyze noise stability of Boolean functions under the noise model where there are two noise operators operating on two parts of the input to the function.

CHAPTER 5

Branching program lower bounds using projective dimension

We study branching program lower bounds using projective dimension, a graph parameter (denoted by $\mathsf{pd}(G)$ for a graph G), introduced by Pudlák and Rödl (1992). For a Boolean function f (on n bits), Pudlák and Rödl associated a bipartite graph G_f and showed that size of the optimal branching program computing f (denoted by $\mathsf{bpsize}(f)$) is at least $\mathsf{pd}(G_f)$ (also denoted by $\mathsf{pd}(f)$). Hence, proving lower bounds for $\mathsf{pd}(f)$ imply lower bounds for $\mathsf{bpsize}(f)$. Despite several attempts (Pudlák and Rödl (1992), Rónyai et.al, (2000)), proving super-linear lower bounds for projective dimension of explicit families of graphs has remained elusive. We observe that there exists a Boolean function f for which the gap between the $\mathsf{pd}(f)$ and $\mathsf{bpsize}(f)$) is $2^{\Omega(n)}$. Motivated by the argument in Pudlák and Rödl (1992), we define two variants of projective dimension - *projective dimension with intersection dimension 1* (denoted by $\mathsf{upd}(f)$) and *bitwise decomposable projective dimension* (denoted by $\mathsf{bpdim}(f)$). We show the following results :

(a) We observe that there exist a Boolean function f for which the gap between $\mathsf{upd}(f)$ and $\mathsf{bpsize}(f)$ is $2^{\Omega(n)}$. In contrast, we also show that the bitwise decomposable projective dimension characterizes size of the branching program up to a polynomial factor. That is, there exists an $0 < \epsilon < 1$ such that for any function f,

$$\mathsf{bitpdim}(f)/6 \leq \mathsf{bpsize}(f) \leq (\mathsf{bitpdim}(f))^{3+\epsilon}$$

(b) We introduce a new candidate function family f for showing super-polynomial lower bounds for $\mathsf{bpdim}(f)$. As our main result, we demonstrate gaps between $\mathsf{pd}(f)$ and the above two new measures for f :

$$\mathsf{pd}(f) = O(\sqrt{n}) \qquad \mathsf{upd}(f) = \Omega(n) \qquad \mathsf{bitpdim}(f) = \Omega\left(\frac{n^{1.5}}{\log n}\right)$$

(c) Although not related to branching program lower bounds, we derive exponential lower bounds for two restricted variants of $\mathsf{pd}(f)$ and $\mathsf{upd}(f)$ respectively by observing that they are equal to well-studied graph parameters - bipartite clique cover number and bipartite partition number respectively.

The results that appear in this chapter are from our works that appear in [KKS16a] and [KKS16b].

5.1 Introduction

In this Chapter we study a combinatorial and linear algebraic approach to proving branching program lower bounds. This approach was proposed by Pudalk and Rodl [PR92] in the 90's. Their idea was to connect branching program size to a linear-algebraic/combinatorial parameter called Projective Dimension. Projective dimension is defined as a measure of bipartite graphs. A natural association between Boolean functions and bipartite graphs can be obtained by partitioning the set of variables into two arbitrary partitions. The projective dimension of a bipartite graph $G(U, V, E)$ over a field $\mathbb{F}$ is the minimum $d, d \in \mathbb{N}$ such that there is an assignment of linear subspaces of $\mathbb{F}^d$ to vertices in both partitions such that for any $(u, v) \in U \times V$ the corresponding subspaces intersect non-trivially if and only if $(u, v) \in E$. Pudlák and Rödl showed that the projective dimension of a Boolean function is upper bounded by deterministic branching program size. Thus, to lower bound branching program size it is enough to lower bound projective dimension.

Pudlák and Rödl [PR92] showed that projective dimension is an interesting parameter by showing that there exists Boolean functions which require exponen-

tial projective dimension. But they do this by a counting argument and hence the resulting lower bound is only for a non-explicit function. But for the goal of separating P from L we need to show a super polynomial lower bound for an explicit function. But the best lower bound on projective dimension for an explicit function that is known is only linear [PR92] in the input length of the function (in other words the logarithm of the number of vertices in the bipartite graph). Also this lower bound is for a function (complement of the Equality function which given two n bit strings checks if they are equal bit by bit) which has linear sized branching program computing it. Thus, by the Pudlák and Rödl connection, this functions projective dimension is at most linear. Hence it is impossible to prove super linear lower bounds on projective dimension of this function.

Though many candidate functions like Payley graphs were suggested [RBG02], it was not clear how to prove a super linear lower bound on projective dimension of any of these functions. We thus looked at variants of projective dimension which are more structured, but still preserve the connection to branching program size. The results we obtain in this chapter are based on our results from [DKS16].

Pudlák and Rödl proved the upper bound of branching program size on projective dimension by constructing a projective dimension assignment from a branching program computing the function. They prove the validity of the assignment constructed by proving that any intersection between the sub-spaces of $u \in U$ and $v \in V$ corresponds to an accepting path in the branching program when the input is uv. Since in a deterministic branching program there is exactly one accepting path on any given input, it is easy to note that this assignment has the additional property that, whenever the subspaces of u and v intersect non-trivially the dimension of the intersection is exactly 1. That is for every $(u, v) \in U \times V$, the dimension

of the intersection is either 0 or 1. We call this variant projective dimension with intersection dimension 1. Though we are able to prove a linear lower bound on this variant for a function whose projective dimension is sub-linear, establishing a quadratic gap between projective dimension and our variant, we were unable to improve it to a super-linear lower bound. This is formalized by the following theorem.

Theorem 5.1.1. *For any $d \geq 0$, for the function SI_d (on $2d^2$ variables, see Definition 5.2.4), the projective dimension is exactly equal to d, while the projective dimension with intersection dimension 1 is $\Omega(d^2)$.*

We then turn to the important question of gaps between projective dimension and branching program size. We answer the question by showing an exponential gap between projective dimension (and even projective dimension with intersection dimension 1) and branching program size. More formally we show the existence (non-explicit, again via a counting argument) of a Boolean function which has linear projective dimension but requires exponential sized branching programs to compute it. This somewhat explains the lack of progress on obtaining a super-linear lower bound for functions which are believed not to have polynomial size branching programs via projective dimension or via the variant of projective dimension we proposed.

Inspired by the gap example, we observe another crucial property of projective dimension assignments constructed by Pudlák and Rödl from the branching program computing the function. This property captures the essence that such a projective dimension assignment is "easy to describe". What we mean by "easy to describe" is that there are $4n$ subspaces, 2 for each of the $2n$ bits such that the assignment for any vertex is a direct sum of n of these subspace based on each bit

of binary input labeling the vertex. We define a variant of projective dimension which captures this property called Bitwise decomposable projective dimension or bitpdim for short. The definition is made such that the connection to branching program size is preserved.

From our definition of bitpdim it is not too hard to see that the gap proof does not apply to bitpdim. We strengthen this intuition by formally proving that there cannot be a super-polynomial gap between it and branching program size. We show that branching program size is upper bounded by $d^{3+\epsilon}$ where d is the bitpdim of the function and $0 \leq \epsilon < 1$ is a constant. Thus, we establish that bitpdim and branching program size are polynomially related. More formally we prove that:

Theorem 5.1.2. *There is an absolute constant $c > 0$ such that if $\mathsf{bitpdim}(f_n) \leq d(n)$ for a function family $\{f_n\}_{n \geq 0}$ on $2n$ bits, then there is a deterministic branching program of size $(d(n))^c$ computing it.*

Our result can also be thought of as an alternative characterization of deterministic branching program size using projective dimension. An interesting outcome of this connection is that any function which is believed to be outside L is a good candidate function for proving super-linear (and even super-polynomial) lower bounds for bitpdim. Because if the bitpdim of such a function turns out to be polynomial it would imply that the function is in L.

We mention one such candidate function and show a super linear lower bound on the bitpdim of the function. But our proof uses the sub-function counting method of Nechiporuk. We would like to prove such a result using more direct algebraic/combinatorial proof than sub-function counting method. We do not yet know how to prove such a result and it is an interesting open problem.

Since our upper bound on branching program by bitpdim loses a factor of 6 in

the exponent we do not get any super linear lower bounds from the best known branching program lower bound [Nec66]. Despite this we are able to show a lower bound matching the best known branching program lower bound on bitpdim, but once again using the sub-function counting idea. We do this by proving the following :

Theorem 5.1.3 (Main Result)**.** *For any $d > 0$,* $\mathsf{bitpdim}(\mathsf{SI}_d)$ *is at least* $\Omega\left(\frac{d^3}{\log d}\right)$.

We remark that Theorem 5.1.3 implies a size lower bound of $\Omega(\frac{n^{1.5}}{\log n})$ for branching programs computing the function SI_d (where $n = d^2$). However, note that this can also be derived from Nechiporuk's method directly applied on branching program size instead of bitpdim. For the Element Distinctness function, the above linear algebraic adaptation of Nechiporuk's method for bitpdim gives $\Omega(\frac{n^2}{\log^2 n})$ lower bounds (for bitpdim and hence for bpsize) which matches with the best lower bound that Nechiporuk's method can derive.

Continuing the quest for better lower bounds for projective dimension, we study two further restrictions. In these variants of pd and upd, the subspaces assigned to the vertices must be spanned by standard basis vectors. We denote the corresponding dimensions as $\mathsf{spd}(f)$ and $\mathsf{uspd}(f)$ respectively. It is easy to see that for any $2n$-bit function, both of these dimensions are upper bounded by 2^n.

We connect these variants to some of the well-studied graph parameters. The *bipartite clique cover number* (denoted by $bc(G)$) is the smallest collection of complete bipartite subgraphs of G such that every edge in G is present in some graph in the collection. If we insist that the bipartite graphs in the collection be edge-disjoint, the measure is called *bipartite partition number* denoted by $bp(G)$. By definition, $bc(G) \leq bp(G)$. These graph parameters are closely connected to communication complexity as well. More precisely, $\log(bc(G_f))$ is exactly the non-deterministic

communication complexity of the function f, and $\log(bp(G_f))$ is a lower bound on the deterministic communication complexity of f (see [Juk12]). In this context, we show the following:

Theorem 5.1.4. *For any Boolean function f, $\mathsf{spd}(f) = bc(G_f)$ and $\mathsf{uspd}(f) = bp(G_f)$.*

Thus, if for a function family, the non-deterministic communication complexity is $\Omega(n)$, then we will have $\mathsf{spd}(f) = 2^{\Omega(n)}$. Thus, both $\mathsf{spd}(\mathsf{DISJ})$ and $\mathsf{uspd}(\mathsf{DISJ})$ are $2^{\Omega(n)}$.

The rest of the Chapter is organized as follows. In Section 5.2 we give definition of projective dimension, make some crucial observations about the projective dimension assignment constructed from the branching program, introduce some Boolean functions which we use in this chapter and also provide a refresher on basic linear algebra needed for this chapter. In Section 5.3, we describe the construction of Pudlák and Rödl and prove the properties of thus constructed projective dimension assignment. We then study projective dimension of graphs under operations like union and intersection in Section 5.4. In Section 5.5 we study the restriction on projective dimension where the dimension of intersections are restricted to be either 0 or 1, prove lower bounds for this variant and also show a quadratic gap between this variant and projective dimension. In the next section, Section 5.6 we show an exponential gap between projective dimension and branching program size. Motivated by this gap example, we then define the bitwise projective dimension in Section 5.7. We first show that bitwise projective dimension is upper bounded by branching program size. Next we show that bitpdim is equal to branching program size up to polynomial factors in Section 5.8. We then proceed to show a lower bound on bitpdim matching the best known branching program lower bound in Section 5.9. We also give a candidate function

for proving super-polynomial lower bounds for bitpdim in Section 5.10. We then show that some restricted variants of projective dimension which are provably not related to branching program size are indeed equivalent to well known graph theoretic parameters in Section 5.11. We conclude the Chapter by discussing the "natural"-ness of projective dimension based lower bound approach.

5.2 Projective dimension

Unless otherwise stated we work over the field $\mathbb{F}_2$. We remark that our arguments do generalize to any finite field. All subspaces that we talk about in this work are linear subspaces. Also $\vec{0}$ and $\{0\}$ denote the zero vector, and zero-dimensional space respectively. For a subspace $U \subseteq \mathbb{F}^n$, we call the ambient dimension of U as n. We denote $e_i \in \mathbb{F}^n$ as the i^{th} standard basis vector with i^{th} entry being 1 and rest of the entires being zero.

We now define Projective dimension of a graph G over a field $\mathbb{F}$, $\mathsf{pd}_{\mathbb{F}}(G)$ formally.

Definition 5.2.1. For a graph $G(U, V, E)$, the projective dimension of G over a field $\mathbb{F}$, denoted by $\mathsf{pd}_{\mathbb{F}}(G)$ (we will drop $\mathbb{F}$ from the notation when it is $\mathbb{F}_2$), is defined as the smallest d for which there is a vector space W of dimension d and a function ϕ mapping vertices in U, V to linear subspaces of W such that for all $(u, v) \in U \times V$, the pair [1] $(u, v) \in E$ if and only if $\phi(u) \cap \phi(v) \neq \{0\}$. A ϕ which satisfy this condition is said to *realize* the graph $G(U, V, E)$.

For a Boolean function $f : \{0,1\}^{2n} \to \{0,1\}$, fix a partition of the input bits into two parts of size n each, and consider the bipartite graph G_f defined on vertex sets $U = \{0,1\}^n$ and $V = \{0,1\}^n$, as $(u, v) \in E$ if and only if $f(uv) = 1$. A ϕ is said to *realize*

[1] It is worth noting that this does not put constraints on vertices u_1, u_2 in the same partitions, i.e., $\phi(u_1) \cap \phi(u_2)$ may or may not have non-trivial overlap.

the function f if it *realizes* G_f. Unless otherwise mentioned, the partition is the one specified in the definition of the function.

We denote by $\mathsf{bpsize}(f)$ the number of vertices (including accept and reject nodes) in the optimal branching program computing f.

Theorem 5.2.2 (Pudlák-Rödl Theorem ([PR92]))**.** *For a Boolean function $f : \{0,1\}^n \to \{0,1\}$ computed by a deterministic branching program of size s and $\mathbb{F}$ being any field and G_f be the graph induced from f by any partition of $[2n]$ into two equal parts, $\mathsf{pd}_{\mathbb{F}}(G_f) \leq s$.*

The proof of this result proceeds by producing a subspace assignment for vertices of G_f from a branching program computing f. We reproduce the proof of the above theorem in our notation, in the next section, and derive the following proposition. See Section 5.3 for the proofs.

Proposition 5.2.3. *For a Boolean function $f : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ computed by a deterministic branching program of size s, there is a collection of subspaces of $\mathbb{F}^s$ denoted $\mathcal{C} = \{U_i^a\}_{i\in[n],a\in\{0,1\}}$ and $\mathcal{D} = \{V_j^b\}_{j\in[n],b\in\{0,1\}}$, where we associate the subspace U_i^a with a bit assignment $x_i = a$ and V_j^b with $y_j = b$ such that if we define the map ϕ assigning subspaces from $\mathbb{F}^s$ to vertices of $G_f(U, V, E)$ where $U = V = \{0,1\}^n$, as $\phi(x) = \underset{1\leq i\leq n}{\mathsf{span}}\{U_i^{x_i}\}$, $\phi(y) = \underset{1\leq j\leq n}{\mathsf{span}}\{V_j^{y_j}\}$, for $x \in U, y \in V$ then the following holds true. Let $S = \{e_i - e_j \mid i, j \in [s], i \neq j\}$.*

1. *for all $(u, v) \in U \times V$, $\phi(u) \cap \phi(v) \neq \{0\}$ if and only if $f(u, v) = 1$.*
2. *for all $(u, v) \in U \times V$, $\mathsf{dim}\left(\phi(u) \cap \phi(v)\right) \leq 1$.*
3. *For any $W \in \mathcal{C} \cup \mathcal{D}$, $\exists S' \subseteq S$ such that $W = \mathsf{span}\{S'\}$.*

We define the following family of functions and family of graphs based on subspaces of a vector space and their intersections.

Definition 5.2.4 ($\mathsf{SI}_d, \mathcal{P}_d$)**.** Let $\mathbb{F}$ be a finite field. Denote by SI_d, the Boolean function defined on $\mathbb{F}^{d\times d} \times \mathbb{F}^{d\times d} \to \{0,1\}$ as for any $A, B \in \mathbb{F}^{d\times d}$ $\mathsf{SI}_d(A,B) = 1$ if and only if $\mathsf{rowspan}(A) \cap \mathsf{rowspan}(B) \neq \{0\}$. Note that the row span is over the field $\mathbb{F}$ (which, in our case, is $\mathbb{F}_2$). Denote by $\mathcal{P}_d$, the bipartite graph (U, V, E) where U and V are the set of all subspaces of $\mathbb{F}^d$. And for any $(I, J) \in U \times V$, $(I, J) \in E \iff I \cap J \neq \{0\}$

We collect the definitions of Boolean functions which we deal with in this work. For $(x, y) \in \{0,1\}^n \times \{0,1\}^n$, $\mathsf{IP}_n(x, y) = \sum_{i=1}^{n} x_i y_i \mod 2$, $\mathsf{EQ}_n(x, y)$ is 1 if $\forall i \in [n]\, x_i = y_i$ and is 0 otherwise, $\mathsf{INEQ}_n(x, y) = \neg\mathsf{EQ}_n(x, y)$ and $\mathsf{DISJ}_n(x, y) = 1$ if $\forall i \in [n]\, x_i \wedge y_i = 0$ and is 0 otherwise. Note that all the functions discussed so far have branching programs of size $O(n)$ computing them and hence have projective dimension $O(n)$ by Theorem 5.2.2.

Let $m \in \mathbb{N}$ and $n = 2m \log m$. The Boolean function, Element Distinctness, denoted ED_n is defined on $2m$ blocks of $2 \log m$ bits, $x_1, \ldots, x_m$ and $y_1, \ldots, y_m$ bits and it evaluates to 1 if and only if all the x_is and y_is take distinct values when interpreted as integers in $[m^2]$. Let q be a power of prime congruent to 1 modulo 4. For $x, y \in \mathbb{F}_q^*$, the Paley function $\mathsf{PAL}_n^q(x, y) = 1$ if $x - y$ is a quadratic residue in $\mathbb{F}_q^*$ and 0 otherwise.

We observe for any induced subgraph H of G, if G is realized in a space of dimension d, then H can also be realized in a space of dimension d. For any $d \in \mathbb{N}$, $\mathcal{P}_d$ appears as an induced subgraph of the bipartite realization of SI_d. Hence, $\mathsf{pd}(\mathsf{SI}_d) \geq \mathsf{pd}(\mathcal{P}_d)$.

5.2.1 Linear algebra basics

We need the following definition of Gaussian coefficients. For non-negative integers n, k and a prime power q, $\begin{bmatrix} n \\ k \end{bmatrix}_q$ is the expression, $\frac{(q^n-1)(q^n-q)\ldots(q^n-q^{k-1})}{(q^k-1)(q^k-q)\ldots(q^k-q^{k-1})}$ if $n \geq k, k \geq 1$, 0 if $n < k, k \geq 1$, 1 if $n \geq 0, k = 0$.

We recall some basic lemmas from linear algebra which we use later. Unless otherwise mentioned, all our algebraic formulations are over finite fields ($\mathbb{F}$ of size q). For vector spaces V_1, V_2 with dimensions k_1, k_2 respectively, thes direct sum $V_1 \oplus V_2$ is the vector space formed by the column space of the matrix $M = \begin{bmatrix} B_1 & 0 \\ 0 & B_2 \end{bmatrix}$ where B_1 is a $k_1 \times k_1$ matrix whose column space forms V_1, B_2 is a $k_2 \times k_2$ matrix whose column space forms V_2. We now state useful properties of direct sums as proportions below. The proof of these propositions, Propositions 5.2.5 and 5.2.6 follows from an exercise problem in [Rom05]. But for sake of completeness we include those proofs here.

Proposition 5.2.5. *For an arbitrary field $\mathbb{F}$, let U_1, V_1 be subspaces of $\mathbb{F}^{k_1}$ and U_2, V_2 be subspaces of $\mathbb{F}^{k_2}$. Then, $(U_1 \oplus U_2) \cap (V_1 \oplus V_2) \neq \{0\} \iff U_1 \cap V_1 \neq \{0\}$ or $U_2 \cap V_2 \neq \{0\}$*

Proof. We will first prove that $(U_1 \oplus U_2) \cap (V_1 \oplus V_2) \neq \{0\}$ implies that there is a vector in the intersection of U_1, V_1 or U_2, V_2. Let w be a non zero vector in $(U_1 \oplus U_2) \cap (V_1 \oplus V_2)$. Since vectors coming from U_1 and V_2 are orthogonal, it must be that there is a vector common in U_1 and V_1 or U_2 and V_2.

Thus, it remains to prove that direct sum preserves intersections of U_1, V_1 and U_2, V_2. Suppose there is a non zero vector $w = (w_1, w_2, \ldots, w_{k_1})^T$ in $U_1 \cap V_1$, then by definition of direct sum, $U_1 \oplus U_2$ and $V_1 \oplus V_2$ will have the common vector $(w_1, w_2, \ldots, w_{k_1}, 0, \ldots, 0)^T \in \mathbb{F}^{k_1+k_2}$. Similar is the case for a vector in $U_2 \cap V_2$. □

Let U, V be two vector spaces. The vector space formed by Span $(\{uv^\top \mid u \in U, v \in V\})$ is called the tensor product of vector spaces U, V, and is denoted as $U \otimes V$. Here u, v are column vectors. A basic fact about tensor product that we need is the following : (See [Hal74, Sec 25]). Let U be a vector space having basis $u_1, u_2, \ldots u_k$ and V be a vector space having basis $v_1, v_2, \ldots, v_\ell$ over some field $\mathbb{F}$ then, vector space $U \otimes V$ has a basis $B = \{u_i v_j^\top \mid i \in \{1, 2, \ldots, k\}, j \in \{1, 2, \ldots, \ell\}\}$ where u, v are column vectors. Hence, for any two vector spaces U, V, $dim(U \otimes V) = dim(U) \times dim(V)$.

Proposition 5.2.6. *For an arbitrary field $\mathbb{F}$, let U_1, V_1 be subspaces of $\mathbb{F}^{k_1}$ and U_2, V_2 be subspaces of $\mathbb{F}^{k_2}$. Then, $(U_1 \otimes U_2) \cap (V_1 \otimes V_2) \neq \{0\} \iff U_1 \cap V_1 \neq \{0\}$ and $U_2 \cap V_2 \neq \{0\}$*

Proof. For the reverse direction, suppose there is a non zero vector w_1 in $U_1 \cap V_1$ and a non zero vector w_2 in $U_2 \cap V_2$, then $w_1^T w_2 \in U_1 \otimes U_2$ and $w_1^T w_2 \in V_1 \otimes V_2$. Hence $w = w_1^T w_2 \in (U_1 \otimes U_2) \cap (V_1 \otimes V_2)$.

For the forward direction, let w be a non zero vector in $(U_1 \otimes U_2) \cap (V_1 \otimes V_2)$. Let $\{e_i\}_{i \in [k_1]}$ be the set of basis vectors for F^{k_1} and $\{\tilde{e}_j\}_{j \in [k_2]}$ be the set of basis vectors for F^{k_2}. Hence for some $\lambda_{ij}, \mu_{ij} \in \mathbb{F}$, w can be written as, $w = \sum_{i,j} \lambda_{ij} e_i^T \tilde{e}_j = \sum_{i,j} \mu_{ij} e_i^T \tilde{e}_j$. Hence, $\sum_{i,j} (\lambda_{ij} - \mu_{ij}) e_i^T \tilde{e}_j = 0$. By linear independence of tensor basis,

$$\lambda_{ij} = \mu_{ij} \; \forall \; (i, j) \in [k_1] \times [k_2] \tag{5.1}$$

Since w is non-zero, there exists i_1, j_1 with $(i_1, j_1) \in [k_1] \times [k_2]$ such that $\lambda_{i_1 j_1} \neq 0$. Applying equation 5.1, we get $\mu_{i_1 j_1} \neq 0$. Hence for (i_1, j_1), $\lambda_{i_1 j_1}, \mu_{i_1 j_1}$ are both non-zero.

Hence it must be that $(U_1 \otimes U_2)$ and $(V_1 \otimes V_2)$ contain the vector $e_{i_1}^T \tilde{e}_{j_1}$. So e_{i_1} must be present in U_1 and V_1 and e_{j_1} must be present in U_2 and V_2 (if not, $e_{i_1}^T \tilde{e}_{j_1}$ would not have appeared in the intersection). Hence $U_1 \cap V_1 \neq \{0\}$ and $U_2 \cap V_2 \neq \{0\}$. □

Let V be a finite dimensional vector space. For any $U \subseteq_S V$, $V = U \oplus U^{\perp}$. Hence for any $v \in V$ there exists a unique $u \in U, w \in U^{\perp}$ such that $v = u + w$. A projection map Π_U is a linear map defined as $\Pi_U(v) = u$ where u is the component of v in U. For any $A, B \subseteq_S V$ with $A \cap B = \{0\}$, let $V = A + B$. Then any vector $w \in V$ can be uniquely expressed as $w = \Pi_A(w) + \Pi_B(w)$. It is easy to see that, for any $A, B \subseteq_S \mathbb{F}^d$, with $A \cap B = \{0\}$, and any $w \in \mathbb{F}^d$, $\Pi_{A+B}(w) = \Pi_A(w) + \Pi_B(w)$.

5.3 Projective dimension and branching program size

In this section, we reproduce the proof of the projective dimension upper bound in terms of branching program size. The proof is originally due to [PR92], but we supply the details which are essential for the additional observations that we make about the projective dimension assignment constructed from the branching program.

A deterministic branching program is a directed acyclic graph G with distinct start (V_0), accept (V_+) and reject (V_-) nodes. Accept and reject nodes have fan-out zero and are called *sink* nodes. Vertices of the DAG, except sink nodes are labeled by variables and have two outgoing edges, one labeled 0 and the other labeled 1. For a vertex labeled x_i, if the input gives it a value $b \in \{0, 1\}$, then the edge labeled b incident to x_i is said to be *closed* and the other edge is *open*. A branching program is said to accept an input x if and only if there is a path from V_0 to V_+ along the closed edges in the DAG. A branching program is said to compute an $f : \{0, 1\}^n \to \{0, 1\}$, if for all $x \in \{0, 1\}^n$, $f(x) = 1$ iff branching program accepts x.

Theorem 5.3.1. *Let $f : \{0, 1\}^{2n} \to \{0, 1\}$ be computed by a branching program $\mathcal{B}$ of size s.*

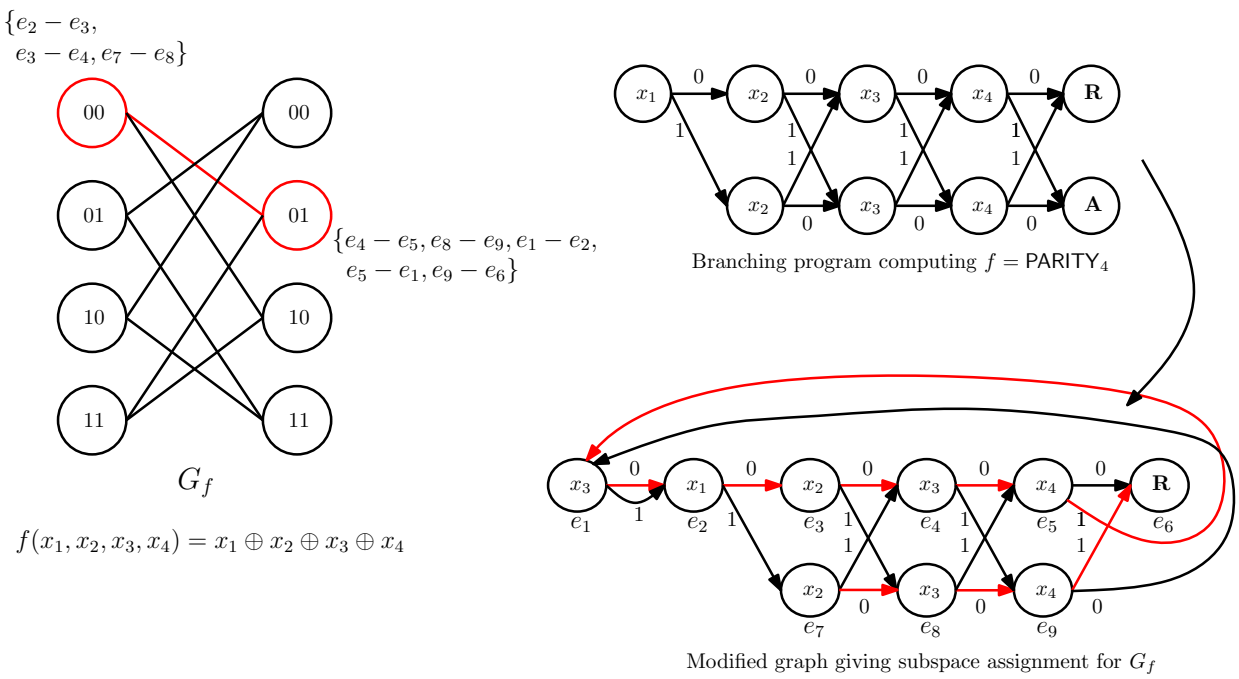


Figure 5.1: Pudlák-Rödl Theorem applied to a branching program computing PARITY_4

Let G_f be the bipartite realization of f, with respect to any partition of $[2n]$ into two equal parts and $\mathbb{F}$ be any arbitrary field. Then, $\mathsf{pd}_{\mathbb{F}}(G_F) \leq s$

Proof. It suffices to come up with a subspace assignment ϕ such that $G_f(P, Q, E)$ has a projective representation in $\mathbb{F}$. Associate u, v to vertices in P, Q respectively. In other words, u corresponds to input variables $\{x_1, x_2, \ldots, x_n\}$ and v corresponds to $\{x_{n+1}, \ldots, x_{2n}\}$ (corresponding to the given partition). By the acceptance property of branching program $\mathcal{B}$, $f(u \circ v) = 1 \iff \exists$ a path from V_0 to accept in $\mathcal{B}$. Since vertices in G_f corresponds to strings in $\{0,1\}^n$, it suffices to give an assignment ϕ such that

$$\exists \text{ a path from start to accept in } \mathcal{B} \iff \text{The bases of } \phi(u), \phi(v) \text{ are linearly dependent} \tag{5.2}$$

We first assign vectors to vertices of the branching program and then use it to come up with a subspace assignment.

Suppose there is a path from V_0 to accept in $\mathcal{B}$. A simple possible way to have dependence is to have the sum of the vectors assigned to the edges of the path telescoping to zero. This can be achieved in the following way.

1. Modify $\mathcal{B}$ by adding a new start vertex labeled with a variable from the other partition from which V_0 got its label. For example, if V_0 is labeled with any of $x_1, x_2, \ldots, x_n$, the new vertex gets its label from $\{x_{n+1}, \ldots, x_{2n}\}$ and vice-versa. Connect both outgoing edges labeled $0, 1$ to V_0.
2. Merge the accept node with the new start node. Let C be the resultant graph which is no longer acyclic. Assign standard basis vectors to each vertex in C.
3. Assign to each edge (u, v) the vector $e_u - e_v$.

Now, the subspace assignment to a vertex $v \in V(G_f)$ is to take the span of all vectors assigned to closed edges on the input v. If there are no closed edges, we assign the zero subspace. With the above modification, cycles in the graph would lead to telescoping of difference vectors (along the cycle edges) to sum to zero.

Modification (1) is necessary as it is possible to have a cycle that does not contain any vertex labeled with $\{x_{n+1}, \ldots, x_{2n}\}$. Then $\phi(v)$ will just be the zero subspace and $\phi(u) \cap \phi(v)$ will be trivial when there is a cycle. It is to avoid this that we add a vertex labeled with a variable from the other partition.

To show that ϕ is a valid subspace assignment, it remains to show that the right-to-left implication of statement 5.2 holds. Suppose for $(u, v) \in E(G_f)$, $\phi(u), \phi(v)$ are linearly dependent. Hence there exists a non trivial combination giving a zero sum.

$$\sum_{\substack{e \in E(C) \\ e=(u,v)}} \lambda_e (e_u - e_v) = 0, \quad \lambda_e \in \mathbb{F} \; \forall e \in E(C)$$

Let S be the non-empty set of edges such that $\lambda_e \neq 0$ and $V(S)$ be its set of vertices. Now for any vertex $u \in V(S)$ there must be at least two edges containing u because with just a single edge ϵ_u, which being a basis vector and summing up to zero,

must have a zero coefficient which contradicts that fact that $e \in S$. This shows that every vertex in S has a degree ≥ 2 (in the undirected sense). Hence it must have an undirected cycle. □

Fig. 5.1 shows the transformations done to the branching program as per the proof of Pudlák-Rödl Theorem and subspace assignment obtained for 00 and 01. The subspace assignment for each of the vertices is listed in the table below.

x_1x_2	Assignment	x_3x_4	Assignment
00	$e_2 - e_3, e_3 - e_4, e_7 - e_8$	00	$e_4 - e_5, e_8 - e_9, e_1 - e_2, e_5 - e_6, e_9 - e_1$
01	$e_2 - e_3, e_3 - e_8, e_7 - e_4$	01	$e_4 - e_5, e_8 - e_9, e_1 - e_2, e_5 - e_1, e_9 - e_6$
10	$e_2 - e_7, e_3 - e_4, e_7 - e_8$	10	$e_4 - e_9, e_8 - e_5, e_1 - e_2, e_5 - e_6, e_9 - e_1$
11	$e_2 - e_7, e_3 - e_8, e_7 - e_4$	11	$e_4 - e_9, e_8 - e_5, e_1 - e_2, e_5 - e_1, e_9 - e_6$

Table 5.1: Subspace assignment for $PARITY_4$ given by proof of Pudlák-Rödl theorem

We now prove the Proposition 5.2.3, establishing the properties of projective dimension assignment constructed from the branching program.

Proof. Proof of Proposition 5.2.3 We reuse the notations introduced in the proof of Theorem 5.2.2 which we have described in 5.3.1. If H_x denotes the set of edges that are closed on an input a, then the subspace assignment $\phi(a)$ is the span of vectors associated with edges of H_x. Denote by $H_{x_i=a_i}$, the subgraph consisting of edges labeled $x_i = a_i$. Hence H_a can be written as span of vectors associated with $H_{x_i=a_i}$. Hence $\phi(a)$ can be expressed as $span_{i=1}^{n} U_i$ where $U_i = span_{(u,v)\in H_{x_i=a_i}}(e_u - e_v)$. A similar argument shows that $\phi(y)$ also has such a decomposition. We now argue the properties of ϕ.

Note that the first and third property directly follow from the construction. To see the second property, observe that the branching program is deterministic and hence there can be only one accepting path. Since we observed that the vectors in

the accepting path contribute to the intersection space and since there is only one such path, the dimension of the intersection spaces is bound to be 1. □

5.4 Projective dimension as property of graphs

In this section, we observe properties about projective dimension as a measure of graphs and Boolean functions. We start by proving closure properties of projective dimension under Boolean operations $\wedge$ and $\vee$. The proof is based on the direct sum and tensor product of vector spaces.

Lemma 5.4.1. *Let $\mathbb{F}$ be an arbitrary field. For any two functions $f_1 : \{0,1\}^{2n} \to \{0,1\}$, $f_2 : \{0,1\}^{2n} \to \{0,1\}$, $\mathsf{pd}_{\mathbb{F}}(f_1 \vee f_2) \leq \mathsf{pd}_{\mathbb{F}}(f_1) + \mathsf{pd}_{\mathbb{F}}(f_2)$ and $\mathsf{pd}_{\mathbb{F}}(f_1 \wedge f_2) \leq \mathsf{pd}_{\mathbb{F}}(f_1) \cdot \mathsf{pd}_{\mathbb{F}}(f_2)$*

Proof. In this proof, for a Boolean f with bipartite representation $G_f(U, V, E)$ we define the map ϕ to be from $\{0,1\}^n \times \{0,1\}$ where $\phi(u,0)$ denotes the subspace assigned to $u \in U$ and $\phi(v,1)$ denotes the subspace assigned to $v \in V$ of G_f. Let f_1 and f_2 be of projective dimensions k_1 and k_2 realized by maps $\phi_1 : \{0,1\}^n \times \{0,1\} \to \mathbb{F}^{k_1}, \phi_2 : \{0,1\}^n \times \{0,1\} \to \mathbb{F}^{k_2}$ respectively.

- From ϕ_1 and ϕ_2 we construct a subspace assignment $\phi : \{0,1\}^n \times \{0,1\} \to \mathbb{F}^{k_1+k_2}$ which realizes $f = f_1 \vee f_2$ thus proving the theorem.
  The subspace assignment is : for $u \in \{0,1\}^n, \phi(u,0) = \phi_1(u,0) \oplus \phi_2(u,0)$. Similarly for $v \in \{0,1\}^n, \phi(v,1) = \phi_1(v,1) \oplus \phi_2(v,1)$. Now, for $u, v \in \{0,1\}^n$, if $f(u,v) = 1$ then it must be that $f_1(u,v) = 1$ or $f_2(u,v) = 1$. Thus, either $\phi_1(u,0) \cap \phi_1(v,1) \neq \{0\}$ or $\phi_2(u,0) \cap \phi_2(v,1) \neq \{0\}$. By Proposition 5.2.5, it must be the case that $(\phi_1(u,0) \oplus \phi_2(u,0)) \cap (\phi_1(v,1) \oplus \phi_2(v,1)) \neq \{0\}$. Hence $\phi(u,0) \cap \phi(v,1) \neq \{0\}$. The dimension of resultant space is $k_1 + k_2$. The case $f(u,v) = 0$ is easy as both sup-spaces in the direct-sum are the trivial subspace $\{0\}$ and so is their direct sum.
- From ϕ_1 and ϕ_2 we construct a subspace assignment $\phi : \{0,1\}^n \times \{0,1\} \to \mathbb{F}^{k_1 k_2}$, realizing $f_1 \wedge f_2$ thus proving the theorem. Consider the following projective dimension assignment ϕ: for $u \in \{0,1\}^n, \phi(u,0) = \phi_1(u,0) \otimes \phi_2(u,0)$. Similarly for $v \in \{0,1\}^n, \phi(v,1) = \phi_1(v,1) \otimes \phi_2(v,1)$. The proof is similar to the previous case and applying Proposition 5.2.6, completes the proof.

□

The $\vee$ part of the above lemma was also observed (without proof) in [PR94]. A natural question is whether we can improve any of the above bounds. In that context, we make the following remarks: (1) the construction for $\vee$ is tight up to constant factors, (2) we cannot expect a general relation connecting $\mathsf{pd}_{\mathbb{R}}(f)$ and $\mathsf{pd}_{\mathbb{R}}(\neg f)$.

- We prove that the construction for $\vee$ is tight up to constant factors. Assume that n is a multiple of 4. Consider the functions $f(x_1, \ldots, x_{\frac{n}{4}}, x_{\frac{n}{2}+1}, \ldots, x_{\frac{3n}{4}})$ and $g(x_{\frac{n}{4}+1}, \ldots, x_{\frac{n}{2}}, x_{\frac{3n}{4}+1}, \ldots, x_n)$ each of which performs inequality check on the first $\frac{n}{4}$ and the second $\frac{n}{4}$ variables. It is easy to see that $f \vee g$ is the inequality function on $\frac{n}{2}$ variables $x_1, \ldots, x_{\frac{n}{2}}$ and the next $\frac{n}{2}$ variables $x_{\frac{n}{2}+1}, \ldots, x_n$. By the fact that they are computed by n size branching programs and using Theorem 5.2.2 (Pudlák-Rödl theorem) we get that $\mathsf{pd}(f) \leq n$ and $\mathsf{pd}(g) \leq n$. Hence by Lemma 5.4.1, $\mathsf{pd}(f \vee g) \leq \mathsf{pd}(f) + \mathsf{pd}(g) \leq 2n$. The lower bound on the projective dimension of the inequality function comes from [PR92, Theorem 4], giving $\mathsf{pd}(f \vee g) \geq \epsilon.\frac{n}{2}$ for an absolute constant ϵ. This shows that $\mathsf{pd}(f \vee g) = \Theta(n)$.
- A natural idea to improve the upper bound of $\mathsf{pd}(f_1 \wedge f_2)$ is to prove upper bounds for $\mathsf{pd}(\neg f)$ in terms of $\mathsf{pd}(f)$. However, we remark that over $\mathbb{R}$, it is known [PR92] that $\mathsf{pd}_{\mathbb{R}}(\mathsf{INEQ}_n)$ is $\Omega(n)$ while $\mathsf{pd}_{\mathbb{R}}(\mathsf{EQ}_n) = 2$. Hence we cannot expect a general relation connecting $\mathsf{pd}_{\mathbb{R}}(f)$ and $\mathsf{pd}_{\mathbb{R}}(\neg f)$.

We now observe a characterization of bipartite graphs having projective dimension at most d over $\mathbb{F}$. Let $f : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$, and $G_f(X, Y, E)$ be its bipartite realization. Let $\mathsf{pd}(G_f) = d$.

Proposition 5.4.2. *For any subspace assignment ϕ realizing G_f, no two vertices from the same partition whose neighborhoods are different can get the same subspace assignment.*

Proof. Suppose there exists $x, x' \in S$ from the same partition, i.e., either X or Y,such that $\phi(x) = \phi(x')$. Since $N(x) \neq N(x')$, without loss of generality, there exists $z \in N(x) \setminus N(x')$. Now since $\phi(x) = \phi(x')$, x' will be made adjacent to z by the

assignment and hence ϕ is no longer a realization of G_f since z should not have been adjacent to x'. □

Lemma 5.4.3 (Characterization)**.** *Let G be a bipartite graph with no two vertices having same neighborhood, $\mathsf{pd}(G) \leq d$ if and only if G is an induced subgraph of $\mathcal{P}_d$.*

Proof. Suppose G appears as an induced subgraph of $\mathcal{P}_d$. To argue, $\mathsf{pd}(G) \leq d$, simply consider the assignment where the subspaces corresponding to the vertices in $\mathcal{P}_d$ are assigned to the vertices of G.

On the other hand, suppose $\mathsf{pd}(G) \leq d$. Let $U_1, \ldots, U_N$ and $V_1, \ldots, V_N$ be subspaces assigned to the vertices. Since the neighborhoods of the associated vertices are different, by Proposition 5.4.2, no two subspaces assigned to these vertices can be the same. Hence corresponding to each vertex in G, there is a unique vertex in $\mathcal{P}_d$ which corresponds to the assignment. Now the subgraph induced by the vertices corresponding to these subspaces in $\mathcal{P}_d$ must be isomorphic to G as the subspace assignment map for G preserves the edge non-edge relations in G. □

It follows that $\mathsf{pd}(\mathcal{P}_d) \leq d$. Observe that, in any projective assignment, the vertices with different neighborhoods should be assigned different subspaces. For $\mathsf{pd}(\mathcal{P}_d)$, all vertices from either partitions have distinct neighborhoods. The number of subspaces of a vector space of dimension $d-1$ is strictly smaller than the number of vertices in $\mathcal{P}_d$. Thus, we conclude the following theorem.

Theorem 5.4.4. *For any $d \in \mathbb{N}$, $\mathsf{pd}(\mathcal{P}_d) = \mathsf{pd}(\mathsf{Sl}_d) = d$.*

For an N vertex graph G, the number of vertices of distinct neighborhood can at most be N. Thus, the observation that we used to show the lower bound for the $\mathsf{pd}(\mathcal{P}_d)$ cannot be used to obtain more than a $\sqrt{\log N}$ (there are at most q^{d^2}

distinct subspaces of dimension d over $\mathbb{F}_q$) lower bound for $\mathsf{pd}(G)$. Also, for many functions, the number of vertices of distinct neighborhood can be smaller.

We observe that by incurring an additive factor of $2\log N$, any graph G on N vertices can be transformed into a graph G' on $2N$ vertices such that all the neighborhoods of vertices in one partition are distinct. Let $f : \{0,1\}^{2n} \to \{0,1\}$ be such that the neighborhoods of G_f are not necessarily distinct. We consider a new function f' whose bipartite realization will have two copies of G_f namely $G_1(A_1, B_1, E_1)$ and $G_2(A_2, B_2, E_2)$ where A_1, A_2, B_1, B_2 are disjoint and a matching connecting vertices in A_1 to B_2 and A_2 to B_1 respectively. Since the matching edge (i.e, the edge from (A_2, B_1) matching or (A_1, B_2) matching) associated with every vertex is unique, the neighborhoods of all vertices are bound to be distinct. Applying Lemma 5.4.1 and observing that matching (i.e, equality function) has projective dimension at most n, $\mathsf{pd}(f') \leq 2\mathsf{pd}(f) + 2n$. This shows that to show super-linear lower bounds on projective dimension for f where the neighborhoods may not be distinct, it suffices to show a super-linear lower bound for f'.

5.5 A restricted variant of projective dimension

Motivated by the proof of Theorem 5.2.2 we make the following definition.

Definition 5.5.1 (Projective Dimension with Intersection Dimension 1). A Boolean function $f : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ with the corresponding bipartite graph $G(U, V, E)$ is said to have projective dimension with intersection dimension 1 (denoted by $\mathsf{upd}(f)$) d over field $\mathbb{F}$, if d is the smallest possible dimension for which there exists a vector space K of dimension d over $\mathbb{F}$ with a map ϕ assigning subspaces of K to $U \cup V$ such that

- for all $(u, v) \in U \times V$, $\phi(u) \cap \phi(v) \neq \{0\}$ if and only if $(u, v) \in E$.
- for all $(u, v) \in U \times V$, $\mathsf{dim}\left(\phi(u) \cap \phi(v)\right) \leq 1$.

By the properties observed in Proposition 5.2.3,

Theorem 5.5.2. *For a Boolean function f computed by a deterministic branching program of size s, $\mathsf{upd}_{\mathbb{F}}(f) \leq s$ for any field $\mathbb{F}$.*

Thus, it suffices to prove lower bounds for $\mathsf{upd}(f)$ in order to obtain branching program size lower bounds.

We now proceed to show lower bounds on upd.

Our approaches use the fact that the adjacency matrix of $\mathcal{P}_d$ has high rank.

Lemma 5.5.3. *Let M be the bipartite adjacency matrix of $\mathcal{P}_d$, then $\mathsf{rank}\,(M) \geq \begin{bmatrix} d \\ d/2 \end{bmatrix}_q \geq q^{\frac{d^2}{4}}$*

Proof. For $0 \leq i \leq k \leq d$, and subspace $I, K \subseteq_s \mathbb{F}_q^d$ with $dim(I) = i, dim(K) = k$, define matrix $\overline{W_{ik}}$ over $\mathbb{R}$ as $\overline{W_{ik}}(I, K) = 1$ if $I \cap K = \{0\}$ and 0 otherwise. This matrix has dimension $\begin{bmatrix} d \\ i \end{bmatrix}_q \times \begin{bmatrix} d \\ k \end{bmatrix}_q$.

Consider the submatrix M_i of M with rows and columns indexed by subspaces of dimension exactly i. Observe that $\overline{W_{ii}} = J - M_i$ where J is an all ones matrix of appropriate order. These matrices are well-studied (see [FW86]). Closed form expressions for eigenvalues are computed in [Del76, LW12] and the eigenvalues are known to be non-zero. Hence for $0 \leq i \leq d/2$ the matrix $\overline{W_{ii}}$ has rank $\begin{bmatrix} d \\ i \end{bmatrix}_q$.

Since $\overline{W_{ii}} = J - M_i$, $\mathsf{rank}\,(M_i) \geq \mathsf{rank}\left(\overline{W_{ii}}\right) - 1$. This shows that $\mathsf{rank}\,(M) \geq \mathsf{rank}\,(M_i) = \begin{bmatrix} d \\ i \end{bmatrix}_q$ for all i such that $2i \leq d$. Choosing $i = d/2$ gives $\mathsf{rank}\,(M) \geq \begin{bmatrix} d \\ d/2 \end{bmatrix}_q - 1 \geq q^{\frac{d^2}{4}} - 1$. □

We now present two approaches for showing lower bounds on $\mathsf{upd}(f)$ - one using intersection families of vector spaces and the other using rectangle arguments on M_f.

Lower Bound for $\mathsf{upd}(\mathcal{P}_d)$ using intersecting families of vector spaces : To prove a lower bound on $\mathsf{upd}(\mathcal{P}_d)$ we define a matrix N from a projective assignment with intersection dimension 1 for $\mathcal{P}_d$, such that it is equal to $(q-1)M$. Let $D = \mathsf{upd}(\mathcal{P}_d)$. We first show that $\mathsf{rank}(N)$ is at most $1 + \begin{bmatrix} D \\ 1 \end{bmatrix}_q$. Then by Lemma 5.5.3 we get that $\mathsf{rank}(N)$ is at least $q^{\frac{d^2}{4}}$. Let $\mathcal{G} = \{G_1, \ldots, G_m\}$, $\mathcal{H} = \{H_1, \ldots, H_m\}$ be the subspace assignment with intersection dimension 1 realizing $\mathcal{P}_d$ with dimension D.

Lemma 5.5.4. *For any polynomial p in q^x of degree s, with matrix N of order $|\mathcal{G}| \times |\mathcal{H}|$ defined as $N[G_r, H_t] = p(\mathsf{dim}(G_r \cap H_t))$ with $G_r \in \mathcal{G}$, $H_t \in \mathcal{H}$, then $\mathsf{rank}(N) \leq \sum_{i=0}^{s} \begin{bmatrix} D \\ i \end{bmatrix}_q$*

Proof. This proof is inspired by the proof in [FG85] of a similar claim where a non-bipartite version of this lemma is proved.

To begin with, note that p is a degree s polynomial in q^x, and hence can be written as a linear combination of polynomials $p_i = \begin{bmatrix} x \\ i \end{bmatrix}_q, 0 \leq i \leq s$. Let the linear combination be given by $p(x) = \sum_{i=0}^{s} \alpha_i p_i(x)$. For $0 \leq i \leq s$ define a matrix N_i with rows and columns indexed respectively by $\mathcal{G}$, $\mathcal{H}$ defined as $N_i[G_r, H_s] = p_i(\mathsf{dim} G_r \cap H_s)$. By definition of N_i, $N = \sum_{i \in [s]} \alpha_i N_i$.

To bound the rank of N_i's we introduce the following families of inclusion matrices. For any $j \in [D]$, consider two matrices Γ_j corresponding to $\mathcal{G}$ and Δ_j corresponding to $\mathcal{H}$ defined as $\Gamma_j(G, I) = 1$ if $\mathsf{dim}(I) = j, G \in \mathcal{G}, I \subseteq_s G$ and 0 otherwise. $\Delta_j(H, I) = 1$ if $\mathsf{dim}(I) = j, H \in \mathcal{H}, I \subseteq_s H$ and 0 otherwise.

Note that the ranks of the these matrices are upper bounded by the number of columns which is $\begin{bmatrix} D \\ j \end{bmatrix}_q$. We claim that for any $i \in \{0, 1, \ldots, s\}$, $\mathsf{rank}(N_i) \leq \begin{bmatrix} D \\ i \end{bmatrix}_q$. This

completes the proof since $N = \sum_{i\in[s]} \alpha_i N_i$.

To prove the claim, let $\mathcal{F}_i$ denote the set of all i dimensional subspace of $\mathbb{F}_q^D$. We show that $N_i = \Gamma_i \Delta_i^T$. Hence $\mathsf{rank}(N_i) \leq \min\{\mathsf{rank}(\Gamma_i), \mathsf{rank}(\Delta_i)\} \leq \begin{bmatrix} D \\ i \end{bmatrix}_q$. For $(G_r, H_t) \in \mathcal{G} \times \mathcal{H}$, $\Gamma_i \Delta_i^T(G_r, H_t) = \sum_{I\in\mathcal{F}_i} \Gamma_i(G_r, I)\Delta_i^T(I, H_t) = \sum_{I\in\mathcal{F}_i} \Gamma_i(G_r, I)\Delta_i(H_t, I) = \sum_{I\in\mathcal{F}_i} [I \subseteq_s G_r] \wedge [I \subseteq_s H_t] = \sum_{I\in\mathcal{F}_i} [I \subseteq_s G_r \cap H_t] = \begin{bmatrix} \dim(G_r\cap H_t) \\ i \end{bmatrix}_q = N_i(G_r, H_t)$ □

We apply Lemma 5.5.4 on N defined via $p(x) = q^x - 1$ with $s = 1$, to get $q^{d^2/4} \leq \begin{bmatrix} d \\ d/2 \end{bmatrix}_q \leq 1 + \begin{bmatrix} D \\ 1 \end{bmatrix}_q = 1 + (q^D - 1)/(q - 1)$. By definition, $\mathsf{rank}(N) = \mathsf{rank}(M)$. This gives that $D = \Omega(d^2)$ and along with Theorem thm:subspace-graph-pd proves Theorem 5.1.1.

Lower Bound for $\mathsf{upd}(\mathcal{P}_d)$ from Rectangle Arguments : We now give an alternate proof of Theorem 5.1.1 using combinatorial rectangle arguments.

Lemma 5.5.5. *For $f : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ with M_f denoting the bipartite adjacency matrix of G_f, $rank_{\mathbb{R}}(M_f) \leq q^{O(\mathsf{upd}_{\mathbb{F}}(f))}$ where $\mathbb{F}$ is a finite field of size q.*

Proof. Let ϕ be a subspace assignment realizing f of dimension d with intersection dimension 1. Let $S(v)$ for $v \in \mathbb{F}_q^d$ denote $\left\{(a,b) \in \{0,1\}^n \times \{0,1\}^n \mid \phi(a) \cap \phi(b) = \mathsf{span}\{v\}\right\}$. Also let M_v denote the matrix representation of $S(v)$. That is, $M_v(a,b) = 1 \iff (a,b) \in S(v)$. Consider all 1 dimensional subspaces which appear as intersection space for some input (x,y). Fix a basis vector for each space and let T denote the collection of basis vectors of all the intersection spaces. Note that for any $(x,y) \in f^{-1}(1)$, there is a unique $v \in \mathbb{F}_q^d$ (up to scalar multiples) such that $(x,y) \in S(v)$ for otherwise intersection dimension exceeds 1. Then $M_f = \sum_{v\in T} M_v$. Now, $rank(M_f) \leq \sum_{v\in T} rank(M_v)$. Since $rank(M_v) = 1$, $rank(M_f) \leq |T|$. The fact that the number of 1 dimensional spaces in $\mathbb{F}^d$ can be at most $\frac{q^d-1}{q-1}$ completes the proof. Note that the rank of M_f can be over any field (we choose $\mathbb{R}$). □

We get an immediate corollary. Any function f, such that the adjacency matrix of M_f of the bipartite graph G_f is of full rank 2^n over some field must have $\mathsf{upd}(f) = \Omega(n)$. There are several Boolean functions with this property, well-studied in the context of communication complexity (see textbook [KN97]). Hence, we have for $f \in \left\{\mathsf{IP}_n, \mathsf{EQ}_n, \mathsf{INEQ}_n, \mathsf{DISJ}_n, \mathsf{PAL}_n^q\right\}$, $\mathsf{upd}_{\mathbb{F}}(f)$ is $\Omega(n)$ for any finite field $\mathbb{F}$.

For arguing about PAL_n^q, it can be observed that the graph is strongly regular (as $q \equiv 1 \mod 4$) and hence the adjacency matrix has full rank over $\mathbb{R}$ [Bol01]. Except for PAL_n^q, all the above functions have $O(n)$ sized deterministic branching programs computing them and hence the Pudlák-Rödl theorem (Theorem 5.2.2) gives that upd for these functions (except PAL_n^q) are $O(n)$ and hence the above lower bound is indeed tight.

From Lemma 5.5.3, it follows that the function SI_d also has rank $2^{\Omega(d^2)}$. To see this, it suffices to observe that $\mathcal{P}_d$ appears as an induced subgraph in the bipartite realization of SI_d. Thus, $\mathsf{upd}(\mathsf{SI}_d)$ is $\Omega(d^2)$. We proved in Theorem 5.4.4 that $\mathsf{pd}(\mathsf{SI}_d) = d$. This establishes a quadratic gap between the two parameters. This completes the alternate proof of Theorem 5.1.1.

Let $D(f)$ denote the deterministic communication complexity of the Boolean function f. We observe that the rectangle argument used in the proof of Lemma 5.5.5 is similar to the matrix rank based lower bound arguments for communication complexity. This yields the Proposition 5.5.6. If $\mathsf{upd}(f) \leq D$, the assignment also gives a partitioning of the 1s in M_f into at most $\frac{q^D-1}{q-1}$ 1-rectangles. However, it is unclear whether this immediately gives a similar partition of 0s into 0-rectangles as well. Notice that if $D(f) \leq d$, there are at most 2^d monochromatic rectangles (counting both 0-rectangles and 1-rectangles) that cover the entire matrix. However, our

proof does not exploit this difference.

Proposition 5.5.6. *For a Boolean function* $f : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ *and a finite field* $\mathbb{F}$, $\mathsf{upd}_{\mathbb{F}}(f) \leq 2^{D(f)}$ *and* $D(f) \leq (\mathsf{pd}_{\mathbb{F}}(f))^2 \log|\mathbb{F}|$

Proof. We give a proof of the first inequality. Any deterministic communication protocol computing f of cost $D(f)$, partitions M_f into k rectangles where $k \leq 2^{D(f)}$ rectangles. Define $f_i : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ for each rectangle R_i $i \in [k]$, such that $f_i(x,y) = 1$ iff $(x,y) \in R_i$. Note that $\mathsf{upd}_{\mathbb{F}}(f_i) = 1$ and $f = \vee_{i=1}^{k} f_i$. For any $(x,y) \in \{0,1\}^n \times \{0,1\}^n$ if $f(x,y) = 1$, there is exactly one $i \in [k]$ where $f_i(x,y) = 1$. Hence for each $j \in [k], j \neq i$, the intersection vector corresponding to the edge (x,y) in the assignment of f_j is trivial. Hence the assignment obtained by applying Lemma 5.4.1, to $f_1, \vee f_2 \vee \ldots f_k$ will have the property that for any (x,y) with $f(x,y) = 1$, the intersection dimension is 1. Hence $\mathsf{upd}_{\mathbb{F}}(f) \leq k \leq 2^{D(f)}$. To prove the second inequality, consider the protocol where Alice sends the subspace associated with her input as a $\mathsf{pd}_{\mathbb{F}}(f) \times \mathsf{pd}_{\mathbb{F}}(f)$ matrix. □

Note that the first inequality is tight, up to constant factors in the exponent. To see this, consider the function $f : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ whose $\mathsf{pd}_{\mathbb{F}}(f) = \Omega(2^{n/2})$ [PR92, Proposition 1] and note that $D(f)$ for any f is at most n. Tightness of the second inequality is witnessed by SI_d since by Lemma 5.5.3 $D(\mathsf{SI}_d) = \Omega(d^2)$ while $\mathsf{pd}(\mathsf{SI}_d) = d$.

5.6 Exponential gap (non-explicit) between projective dimension and BP Size

The restriction of intersection dimension being 1, although potentially useful for lower bounds for branching program size, does not capture the branching program size exactly. We start the section by demonstrating a function where the gap is exponential.

We show the existence of a Boolean function f such that the size of the optimal branching program computing it is very high but has a very small projective assignment with intersection dimension 1 for any balanced partition of the input.

Proposition 5.6.1. *(Implicit in Remark 1.30 of [Juk12]) There exist a function $f : \{0,1\}^n \times \{0,1\}^n$ that requires size $\Omega(\frac{2^n}{n})$ for any branching program computing f but the $\mathsf{upd}(f) \leq O(n)$ for any balanced partitioning of the input into two parts.*

Proof. Consider the function EQ_n. The graph $G_{\mathsf{EQ}_n}(U, V, E)$ with $U = V = N$ is a perfect matching where $N = \{0,1\}^n$. Relabel the vertices in U of this graph to produce a family $\mathcal{G}$ of $N!$ different labeled graphs. Let $\mathcal{F}$ be the set of Boolean functions whose corresponding graph is in $\mathcal{G}$. Let t be the smallest number such that any function in $\mathcal{F}$ can be computed by a branching program of size at most t. The number of branching programs of size $\leq t$ (bounded by $O(t^t)$ [Juk12]) forms an upper bound on $|\mathcal{F}|$. Thus, $2^{O(t \log t)} \geq N!$, and hence t is $\Omega\left(\frac{2^n}{n}\right)$. Hence there must exist a function $f \in \mathcal{F}$ such that $\mathsf{upd}(f) = \mathsf{upd}(\mathsf{EQ}_n) \leq n$ but $\mathsf{bpsize}(f)$ is $\Omega\left(\frac{2^n}{n}\right)$ for this partition.

We now argue the upper bound for $\mathsf{upd}(f)$ for any balanced partition. Consider the function f_π obtained by a permutation $\pi \in S_N$ on the U part of EQ_n graph.

Consider a partition Π of $[2n]$. Let $G^{\Pi}_{\mathsf{EQ}_n}, G^{\Pi}_{f_\pi}$ be the corresponding bipartite graphs (and EQ^{Π}_n and f^{Π}_π be the corresponding functions) with respect to the partition Π, of EQ_n and f_π respectively.

We claim that $\mathsf{upd}(G^{\Pi}_{\mathsf{EQ}_n}) = \mathsf{upd}(G^{\Pi}_{f_\pi})$.

By definition for any $(u, v) \in \{0,1\}^n \times \{0,1\}^n$, $f_\pi(u, v) = \mathsf{EQ}_n(\pi^{-1}(u), v)$. Also, let (u', v') be the corresponding inputs according to the partition Π of $[2n]$. That is $f^{\Pi}_\pi(u', v') = f_\pi(u, v) = \mathsf{EQ}_n(\pi^{-1}(u), v)$. Let $x = \pi^{-1}(u)$ and $y = v$. Observe that, for $(x, y) \in \{0,1\}^n \times \{0,1\}^n$ there is a unique (x', y') corresponding to it. Hence $f^{\Pi}_\pi(u', v') = \mathsf{EQ}_n(\pi^{-1}(u), v) = \mathsf{EQ}^{\Pi}_n(x', y')$. Thus, for any input (u', v') of f^{Π}_π there is unique input (x', y') of EQ^{Π}_n obtained via the above procedure. Thus, from the upd assignment for EQ^{Π}_n we can get a upd assignment for f^{Π}_π. Observing that Theorem 5.5.2 holds for any partition Π of the input, we get a upd assignment for EQ^{Π}_n. □

The above proposition can be shown by adapting the counting argument presented in Remark 1.30 of [Juk12].

5.7 Bitwise projective dimension (BitPdim)

Motivated by the strong properties observed in Proposition 5.2.3, we make the following definition.

Definition 5.7.1 (Bitwise Decomposable Projective Dimension). Let f be a Boolean function on $2n$ bits and G_f be its bipartite realization. The bipartite graph $G_f(X, Y, E)$ is said to have *bit projective dimension*, $\mathsf{bitpdim}(G) \leq d$, if there exists a collection of subspaces of $\mathbb{F}_2^d$ denoted $\mathcal{C} = \{U^a_i\}_{i\in[n],a\in\{0,1\}}$ and $\mathcal{D} = \{V^b_j\}_{j\in[n],b\in\{0,1\}}$ where a projective

assignment ϕ is obtained by associating subspace U_i^a with a bit assignment $x_i = a$ and V_j^b with $y_j = b$ satisfying the conditions listed below.

1. for all $(x, y) \in \{0,1\}^n \times \{0,1\}^n$, $\phi(x) = \operatorname{span}_{1 \le i \le n}\{U_i^{x_i}\}$, $\phi(y) = \operatorname{span}_{1 \le j \le n}\{V_j^{y_j}\}$ and f is realized by ϕ.
2. Let $S = \{e_i - e_j \mid i, j \in [d], i \neq j\}$. For any $W \in \mathcal{C} \cup \mathcal{D}$, $\exists S' \subseteq S$ such that $W = \operatorname{span}\{S'\}$.
3. for any $S_1, S_2 \subseteq ([n] \times \{0,1\})$ such that $S_1 \cap S_2 = \phi$, $\operatorname{span}_{(i,a) \in S_1}\{U_i^a\} \cap \operatorname{span}_{(j,b) \in S_2}\{U_j^b\} = \{0\}$. The same property must hold for subspaces in $\mathcal{D}$.

We show that the new parameter bitwise decomposable projective dimension (bitpdim) tightly characterizes the branching program size, up to constants in the exponent.

Lemma 5.7.2. *Suppose $f : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ has deterministic branching program of size s then $\operatorname{bitpdim}(f) \le 6s$.*

Proof. The subspace assignment obtained by applying (Theorem 5.3.1) on an arbitrary branching program need not satisfy Property 3 because there can be a vertex z that has two edges incident on it reading different variables from the same partition. To avoid this, we subdivide every edge. We show that this transformation is sufficient to get a bitpdim assignment. We now give a full proof. Let B be a deterministic branching program computing f. Denote the first n variables of f as x and the rest as y. We first apply the Pudlák-Rödl transformation on B to obtain a branching program B' computing f. We note that $|V(B')| = |V(B)|$. Obtain B'' from B' by subdividing every edge (u, v) checking a variable $x_i = b$ from partition x to get three edges, (u, V_{uv}) checking $x_i = b$ and two edges between (V_{uv}, v) one which checks $y_1 = 0$ and another which checks $y_1 = 1$ (see Figure 5.2).

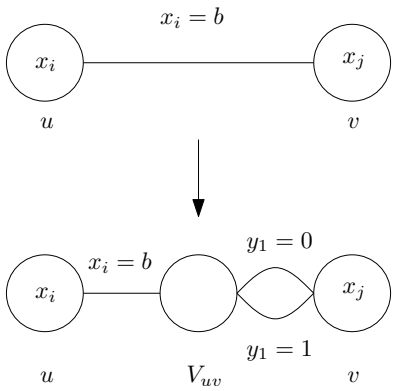


Figure 5.2: Edge modification

Clearly the transformation does not change the function computed by the branching program. Since we are taking every edge of the branching program B' and introducing two more edges, the total number of edges in B' is $3|E(B')|$. Since B' is a deterministic branching program, every vertex $v \in B'$ has out degree at most 2 and at least 1 for every node except sink node. Hence $|E(B')| \leq 2(|V(B')|)$. Along with $|E(B'')| = 3|E(B')|$, we get $|E(B'')| \leq 6(|V(B')|) = 6(|V(B)|)$. Now label every vertex of B'' with standard basis vectors as it is done in the Pudlák-Rödl Theorem (Theorem 5.3.1). Let ϕ be projective assignment obtained from B'' via Pudlák-Rödl theorem. We claim that ϕ satisfies all the requirements of $\mathsf{bitpdim}(f)$.

1. Since ϕ is obtained via Pudlák-Rödl it captures adjacencies of G_f. Hence property 1 holds. Property 2 is satisfied by Pudlák-Rödl assignment.
2. The standard basis vector e_u corresponding to vertex u appears only in edges incident on u in the Pudlák-Rödl assignment. For any edge (u, v) querying a variable $x_i = b$ the other edges incident to v must query variables from y. All the edges incident on u, except (u, v) must also query variables from y. Otherwise, there is an edge (w, u) which queries a variable x_j and our transformation would have subdivided the edge. Hence e_u, e_v belong only to $H_{x_i=b}$ amongst $\{H_{x_i=b}\}_{i\in[n],b\in\{0,1\}}$. This implies Property 3.

□

5.8 BitPdim is equivalent to Branching program size up to polynomial factors

We show that given a bitpdim assignment for a function f, we can construct a branching program computing f.

Theorem 5.8.1 (Theorem 5.1.2 restated)**.** *There is an absolute constant $c > 0$ such that if $\mathsf{bitpdim}(f_n) \leq d(n)$ for a function family $\{f_n\}_{n\geq 0}$ on $2n$ bits, then there is a deterministic branching program of size $(d(n))^c$ computing it.*

Proof. Consider the subspace associated with the variables $\mathcal{C}, \mathcal{D}$ of the bitpdim assignment as the advice string. These can be specified by a list of n basis matrices each of size d^2. Since $d = \mathsf{bitpdim}(f) = poly(n)$, the advice string is $poly(n)$ sized and depends only on n.

We construct a deterministic branching program computing f as follows. On input x, y, from the basis matrices in $\mathcal{C}, \mathcal{D}$, construct an undirected graph[2] G^* with all standard basis vectors in $\mathcal{C}, \mathcal{D}$ as vertices and add an edge between two vertices u, v if $e_u - e_v \in U_i^{x_i}$ or $e_u - e_v \in V_j^{y_j}$ for $i, j \in [n]$. For input x, y, $f(x, y) = 1$ iff G^* has a cycle. To see this, let $C = C_1 \cup C_2$ be a cycle in G^* where C_1 consists of edges from basis matrices in $\mathcal{C}$ and C_2 contain edges from basis matrices in $\mathcal{D}$. Note that if one of C_1 or C_2 is empty then there is a cycle consisting only of vectors from $\mathcal{C}$ which implies a linear dependence among vectors in $\mathcal{C}$. But this contradicts Property 3 of bitpdim assignment. Hence both C_1 and C_2 are non-empty.

Then it must be that $\sum_{(u,v)\in C_1} e_u - e_v \neq 0$, $\sum_{(w,z)\in C_2} e_w - e_z \neq 0$ and $\sum_{(u,v)\in C_1} e_u - e_v + \sum_{(w,z)\in C_2} e_w - e_z = 0$. Hence $\sum_{(u,v)\in C_1} e_u - e_v = -\sum_{(w,z)\in C_2} e_w - e_z$. Hence we get a vector in the intersection which gives $f(x, y) = 1$. Note that if $f(x, y) = 1$, then

[2]Note that this is not a deterministic branching program.

clearly there is a non-zero intersection vector. If we express this vector in terms of the basis, we get a cycle in G^*.

Hence, to check if f evaluates to 1, it suffices check if there is a cycle in G^* which is solvable in L using Reingold's algorithm [Rei08]. The log-space algorithm can also be converted to an equivalent branching program of size n^c for a constant c.

We can improve the constant c to $3 + \epsilon$. We achieve this using the well known random walk based RL algorithm for reachability [AKL$^+$79], amplifying the error and suitably fixing the random bits to achieve a non-uniform branching program of size $d^{3+\epsilon}$.

The RL algorithm requires to store $\log d$ bits to remember the current vertex while doing the random walk and another $\log d$ bits to store the next vertex in the walk. It performs a walk of length $4d^3$ and answers correctly with probability of $1/2$ [MU05]. Amplifying the error does not incur any extra space as the algorithm has a one-sided error and it never errs when it accepts. This gives a probabilistic Turing machine using $2 \log d + 1$ work space. By amplifying the success probability, we can obtain a choice of random bits which works for all inputs of a fixed length. The conversion of this machine to a branching program will incur storing of the head index position of the work tape and input tape position which incur an additional $\log \log d + \log d$ space. Hence overall space is $3 \log d + \log \log d = (3 + \epsilon) \log d$ for small fixed $\epsilon > 0$, thus proving that $c \leq 3 + \epsilon$.

□

5.9 Superlinear lower bound for BitPdim matching the best BP size lower bound

From the results of the previous section, it follows that size lower bounds for branching programs do imply lower bounds for bitwise decomposable projective dimension as well. As mentioned earlier, the lower bounds that Theorem 5.1.2 can give for bitwise decomposable projective dimension are only known to be sub-linear.

To prove super-linear lower bounds for bitwise decomposable projective dimension, we show that Nechiporuk's method [Nec66] can be adapted to our linear algebraic framework (thus proving Theorem 5.1.3). The overall idea is the following: given a function f and a bitpdim assignment ϕ, consider the restriction of f denoted f_ρ where ρ fixes all variables except the ones in T_i to 0 or 1 where T_i is some subset of variables in the left partition. For different restrictions ρ, we are guaranteed to get at least some number $c_i(f)$ of different functions. We show that for each restriction ρ, we can obtain an assignment from ϕ realizing f_ρ. Hence the number of different bitpdim assignments for ρ restricted to T_i is at least the number of sub functions of f which is at least $c_i(f)$. Let d_i be the ambient dimension of the assignment when restricted to T_i. By using the structure of bitpdim assignment, we count the number of assignments possible and use this relation to get a lower bound on d_i. Now repeating the argument with disjoint T_i, and by observing that the subspaces associated with T_is are disjoint, we get a lower bound on d as $d = \sum_i d_i$.

Theorem 5.9.1. *For a Boolean function $f : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ on $2n$ variables, let $T_1, \ldots, T_m$ be a partition of variables into m blocks of size r_i on the first n variables. Let*

$c_i(f)$ be the number of distinct sub functions of f when restricted to T_i, then $\mathsf{bitpdim}(f) \geq \sum_{i=1}^{m} \frac{\log c_i(f)}{\log(\log c_i(f))}$

Proof. Let (x, y) denote the $2n$ input variables of f and $\rho : \{x_1, \ldots, x_n, y_1, \ldots, y_n\} \to \{0, 1, *\}$ be a map that leaves only variables in T_i unfixed. Let ϕ be a bitpdim assignment realizing f and let $G_f(X, Y, Z)$ denote the bipartite realization of f. Let $\mathcal{C} = \left\{U_i^a\right\}_{i\in[n],a\in\{0,1\}}, \mathcal{D} = \{V_j^b\}_{j\in[n],b\in\{0,1\}}$ be the associated collection of subspaces. Let ρ be a restriction that does not make f_ρ a constant and $(x, y) \in \{0, 1\}^n \times \{0, 1\}^n$ which agrees with ρ. We use x, y to denote both variables as well as assignment. From now on, we fix an i and a partition T_i.

Define $L = \underset{i\in[n],\rho(i)\neq *}{\mathsf{span}} \{U_i^{\rho(i)}\}$ and $R = \underset{j\in[n]}{\mathsf{span}}\{V_j^{\rho(n+j)}\}$. For any $x \in \{0, 1\}^n$ that agrees with ρ on the first n bits, define $Z^x = \underset{j\in T_i}{\mathsf{span}}\{U_j^{x_i}\}$. Note that any (x, y) which agrees with ρ has $\phi(x) = L + Z^x$ and $\phi(y) = R$. For any $f_{\rho_1} \not\equiv f_{\rho_2}$, $G_{f_{\rho_1}} \neq G_{f_{\rho_2}}$. Hence the number of bitpdim assignments is at least the number of different sub functions. We need to give a bitpdim assignment for $G_{f_\rho}(V_1, V_2, E)$ where $V_1 = \{x \mid x \text{ agrees with } \rho\}$, $V_2 = \{y\}$ where $y = \rho_{[n+1,\ldots,2n]}$ and $E = \{(x, y) | x \in V_1, y \in V_2, f(x, y) = 1\}$. We use the following property to come up with such an assignment.

Property 5.9.2. *Let ρ be a restriction which does not make the function f constant and which fixes all the variables $y_1, \ldots, y_n$. For all such ρ and $\forall x, y \in \{0, 1\}^n$ which agrees with ρ, any non-zero $w \in \phi(x) \cap \phi(y)$, where $w = u + v$ with $u \in L$ and $v \in Z^x$ must satisfy $v \neq \vec{0}$.*

Proof. Suppose there exists an intersection vector $w \in (L + Z^x) \cap R$ with $w = u + v$, $u \in L$ and $v \in Z^x$ and $v = \vec{0}$. Since $\vec{0} \in Z^{\hat{x}}$ for any $\hat{x}$, $w = u + \vec{0}$ is in $L + Z^{\hat{x}}$ and R. Thus, the function after restriction ρ is a constant. This contradicts the choice of ρ. □

The assignment ψ_ρ for G_{f_ρ} is defined as : $\psi_\rho(x) = Z^x$ and $\psi_\rho(y) = \underset{x \in V_1}{\mathsf{span}}\{\Pi_{Z^x}(R \cap (L + Z^x))\}$
Note that for $(x, y) \in V_1 \times V_2$, $f_\rho(x) = f(x, y)$. The following claim shows that ψ_ρ realize f_ρ.

Claim 5.9.3. *For any $(x, y) \in V_1 \times V_2$, $f(x, y) = 1$ if and only if $\psi_\rho(x) \cap \psi_\rho(y) \neq \{0\}$.*

Proof. For any $(x, y) \in X \times Y$, $\phi(x) \cap \phi(y) \neq \{0\}$ if and only if $f(x, y) = 1$. Since $V_1 \subseteq X$ and $V_2 \subseteq Y$, it suffices to prove : $\forall (x, y) \in V_1 \times V_2$, $\psi_\rho(x) \cap \psi_\rho(y) \neq \{0\} \iff \phi(x) \cap \phi(y) \neq \{0\}$.

We first prove that $\psi_\rho(x) \cap \psi_\rho(y) \neq \{0\}$ implies $\phi(x) \cap \phi(y) \neq \{0\}$. Let v be a non-zero vector in $\psi_\rho(x) \cap \psi_\rho(y)$. By definition of $\psi_\rho(x)$, $v \in Z^x$. By definition of $\psi_\rho(y)$, there exists a non-empty $J \subseteq V_1$ such that $v = \sum_{\hat{x} \in J} v_{\hat{x}}$ where $v_{\hat{x}} \in Z^{\hat{x}}$. Also for every $\hat{x} \in J$, there exists a $u_{\hat{x}} \in L$ such that $w_{\hat{x}} = u_{\hat{x}} + v_{\hat{x}}$ and $w_{\hat{x}} \in R$. Define u to be $\sum_{\hat{x} \in J} u_{\hat{x}}$. Since each $u_{\hat{x}}$ is in L, u is also in L. Hence $w = u + v$ is in $L + Z^x$. Substituting u with $\sum_{\hat{x} \in J} u_{\hat{x}}$ and v with $\sum_{\hat{x} \in J} v_{\hat{x}}$ we get that $w = \sum_{\hat{x} \in J} u_{\hat{x}} + v_{\hat{x}} = \sum_{\hat{x} \in J} w_{\hat{x}}$. Since each $w_{\hat{x}} \in R$, $w \in R$. Hence $w \in R \cap (L + Z^x)$ and w is non-zero as J is non-empty.

Now we prove that $\phi(x) \cap \phi(y) \neq \{0\}$ implies $\psi_\rho(x) \cap \psi_\rho(y) \neq \{0\}$. Let w be non zero vector in $\phi(x) \cap \phi(y)$ with $w = u + v$ where $u \in L$ and $v \in Z^x$. By Property 5.9.2 we have $v \neq \vec{0}$. By definition $v \in \psi_\rho(y)$. Along with $v \in Z^x$, we get $\psi_\rho(x) \cap \psi_\rho(y) \neq \{0\}$. □

Let $Z = \underset{j \in T_i}{\mathsf{span}}\{U_j^0 + U_j^1\}$. We now prove that subspace assignment on the only vertex in the right partition of G_ρ which is $\underset{x \in V_1}{\mathsf{span}}\{\Pi_{Z^x}(R)\}$ is indeed $\Pi_Z(R)$.

Claim 5.9.4. $\Pi_Z(R) = \underset{x \in V_1}{\mathsf{span}}\{\Pi_{Z^x}(R)\}$

Proof. We show $\underset{x \in V_1}{\mathsf{span}}\{\Pi_{Z^x}(R)\} \subseteq \Pi_Z(R)$. Note that $\underset{x \in V_1}{\mathsf{span}}\{\Pi_{Z^x}(R)\} = \underset{x \in V_1, w \in R}{\mathsf{span}}\{\Pi_{Z^x}(w)\}$. For an arbitrary $x \in V_1$ and $w \in R$, let $v = \Pi_{Z^x}(w)$. By definition of Z^x and the fact

that $\left\{U_i^b\right\}_{i\in[n],b\in\{0,1\}}$ are disjoint, $\Pi_{Z^x}(w) = +_{i\in[n],\rho(i)=*}\Pi_{U_i^{x_i}}(w)$. As $Z = \underset{j\in T_i}{\mathsf{span}}\{U_j^0 + U_j^1\}$, every $\Pi_{U_i^{x_i}}(w) \in \Pi_Z(R)$. Hence the span is also in $\Pi_Z(R)$.

Now we show that $\Pi_Z(R) \subseteq \underset{x\in V_1}{\mathsf{span}}\{\Pi_{Z^x}(R)\}$. Let $T_i = \{i_1, \ldots, i_k\}$. For $1 \le j \le k$ define x^j to be $x + e^j$ where $x \in \{0,1\}^n$ agrees with ρ and for any index $i \in [n]$ with $\rho(i) = *$, $x_i = 0$ and $e^j \in \{0,1\}^n$ is 0 at every index other than i_j. Note that for any $j_1 \neq j_2, j_1, j_2 \in T_i$, $Z^{x^{j_1}} \cap Z^{x^{j_2}} = \{0\}$ by Property 3 of Definition 5.7.1) Also note that $\underset{j\in T_i}{\mathsf{span}}\{Z^{x^j}\} = \underset{j\in T_i}{\mathsf{span}}\{U_j^{x_j}\} = Z$. Hence, $\Pi_Z(R) = \underset{j\in T_i}{\mathsf{span}}\{\Pi_{Z^{x^j}}(R)\}$. But $\underset{j\in T_i}{\mathsf{span}}\{\Pi_{Z^{x^j}}(R)\} \subseteq \underset{x\in V_1}{\mathsf{span}}\{\Pi_{Z^x}(R)\}$. □

For any ρ, which fixes all variables outside T_i, Z is the same. And since there is only one vertex on the right partition, for different ρ, ρ', $\Pi_Z(R_\rho) = \Pi_Z(R_{\rho'})$ implies $\psi_\rho = \psi_{\rho'}$. Hence to count the number of different ψ_ρ's for different f_ρ's it is enough to count the number of different $\Pi_Z(R)$. To do so, we claim the following property on $\Pi_Z(R)$.

Property 5.9.5. *Let $S = \{e_u - e_v | e_u - e_v \in Z\}$. Then there exists a subset S' of S such that all the vectors in S' are linearly independent and $\Pi_Z(R) = \mathsf{span}\,\{S'\}$.*

Proof. By the property of the bitpdim assignment, $\forall i \in [n]$ and $\forall b \in \{0,1\}$, $V_i^b = \mathsf{span}\left\{F_i^b\right\}$ where F_i^b is a collection of difference of standard basis vectors. Recall that $R = \underset{j\in[n]}{\mathsf{span}}\{V_j^{\rho(n+j)}\}$. Let $F = \left\{(e_u - e_v) \mid e_u - e_v \in F_j^{\rho(n+j)}, j \in [n]\right\}$. Since projections are linear maps and the fact that $F_j^{\rho(n+j)}$ spans $V_j^{\rho(n+j)}$ we get that, $\Pi_Z(R) = \underset{w\in F}{\mathsf{span}}\{\Pi_Z(w)\}$. Since Z is also a span of difference of standard basis vectors, $\Pi_Z(e_u - e_v)$ is one of $\vec{0}$, $e_u - e_w$ or $e_w - e_v$ where e_w is some standard basis vector in Z. Let $S'' = \cup_{e_u-e_v\in F}\Pi_Z(e_u - e_v)$. Hence $S'' \subseteq S$. Clearly, $\underset{e_u-e_v\in S''}{\mathsf{span}}\{e_u - e_v\} = \Pi_Z(R)$. Choose S' as a linear independent subset of S''. □

Property 5.9.5 along with the fact that $\Pi_Z(R)$ is a subspace of Z, gives us that

the number of different $\Pi_Z(R)$ is upper bounded by number of different subsets S' of S such that $|S'| = d_i$ where $d_i = \mathsf{dim}(Z)$. As S' is a set of difference of standard basis vectors from Z, $|S'| \leq \binom{d_i}{2}$. Thus, the number of different such S' are at most $\sum_{k=0}^{d_i} \binom{d_i^2}{k} = 2^{O(d_i \log d_i)}$.

Hence the number of restrictions ρ (that leaves T_i unfixed) and leading to different f_ρ is at most $2^{O(d_i \log d_i)}$. But the number of such restrictions ρ is at least $c_i(f)$. Hence $2^{O(d_i \log d_i)} \geq c_i(f)$ giving $d_i = \Omega\left(\frac{\log c_i(f)}{\log(\log c_i(f))}\right)$. Using $d = \sum_i d_i$ completes the proof. □

Theorem 5.9.1 gives a super linear lower bound for Element Distinctness function. From a manuscript by Beame et.al, ([BGMS16], see also [Juk12], Chapter 1), we have $c_i(ED_n) \geq 2^{n/2}/n$. Hence applying this count to Theorem 5.9.1, we get that $d \geq \Omega\left(\frac{n}{\log n} \cdot \frac{n}{\log n}\right) = \Omega\left(\frac{n^2}{(\log n)^2}\right)$.

Now we apply this to our context.

To get a lower bound using framework described above it is enough to count the number of sub-functions of Sl_d.

Lemma 5.9.6. *For any $i \in [d]$, there are $2^{\Omega(d^2)}$ different restrictions ρ of Sl_d which fixes all entries other than ith row of the $d \times d$ matrix in the left partition.*

Proof. Fix any $i \in [d]$. Let S be a subspace of $\mathbb{F}_2^d$. Define ρ_S to be $\mathsf{Sl}_d(\mathbf{A}, B)$ where B is a matrix whose rowspace is S. And $\mathbf{A}$ is the matrix whose all but ith row is 0's and ith row consists of variables $(x_{i_1}, \ldots, x_{i_n})$. Thus, for any $v \in \{0,1\}^d$, rowspace of $\mathbf{A}(x)$ is $\mathsf{span}\,\{v\}$.

We claim that for any $S, S' \subseteq_S \mathbb{F}_2^d$ where $S \neq S'$, $(\mathsf{Sl}_d)_{\rho_S} \not\equiv (\mathsf{Sl}_d)_{\rho'_S}$. By definition $(\mathsf{Sl}_d)_{\rho_S} \equiv \mathsf{Sl}_d(\mathbf{A}, B)$ and $(\mathsf{Sl}_d)_{\rho'_S} \equiv \mathsf{Sl}_d(\mathbf{A}, B')$ where B and B' are matrices whose

rowspaces are S and S' respectively. Since $S \neq S'$ there is at least one vector $v \in \mathbb{F}_2^d$ such that it belongs to only one of S, S'. Without loss of generality let that subspace be S. Then $\mathsf{SI}_d(\mathbf{A}(v), B) = 1$ as $v \in S$ where as $\mathsf{SI}_d(\mathbf{A}(v), B') = 0$ as $v \notin S'$. Hence the number of different restrictions is at least number of different subspaces of $\mathbb{F}_2^d$ which is $2^{\Omega(d^2)}$. Hence the proof. □

This completes the proof of Theorem 5.1.3. This implies that for SI_d, the branching program size lower bound is $\Omega\left(\frac{d^2}{\log d} \times d\right) = \Omega\left(\frac{d^3}{\log d}\right) = \Omega\left(\frac{n^{1.5}}{\log n}\right)$ where $n = 2d^2$ is the number of input bits of SI_d.

5.10 A candidate function for P vs L via BitPdim

Assuming $\mathsf{C_=L} \not\subseteq \mathsf{L/poly}$, the function SI_d (a language which we will show is hard for $\mathsf{C_=L}$ under Turing reductions) cannot be computed by deterministic branching programs of polynomial size.

Proposition 5.10.1. *The function family $\{\mathsf{SI}_d\}_{d\geq 0}$ is hard for $\mathsf{C_=L}$ via logspace Turing reductions. Moreover, the negation of $\{\mathsf{SI}_d\}_{d\geq 0}$ is in $\mathsf{L}^{\mathsf{C_=L}}$ (and hence in NC^2).*

Proof. We start with the following fact.

Fact 5.10.2 (Corollary 2.3 of [ABO99])**.** *Fix an $n \in \mathbb{N}$. There exists a logspace computable function $g : \mathbb{F}^{n\times n} \to \mathbb{F}^{n\times n}$ such that for any matrix M over $\mathbb{F}^{n\times n}$, $det(M) = 0 \implies rank(g(M)) = n$ and $det(M) \neq 0 \implies rank(g(M)) = n - 1$*

Consider the language $L = \left\{(M_1, M_2) \mid \mathsf{rowspan}(M_1) \cap \mathsf{rowspan}(M_2) \neq \{0\}, M_1, M_2 \in \mathbb{F}^{d\times d}\right\}$. The reduction is as follows. Given an $M \in \mathbb{F}^{d\times d}$, apply g (defined in Fact 5.10.2) on M to get N, and define for $1 \leq i \leq d$, $H^i = (M_1^i, M_2^i)$ where M_1^i is the matrix

consisting of the i^{th} row of N repeated n times and M_2^i is the same as N with the i^{th} row replaced by all 0 vector. For each $1 \leq i \leq d$, we make an oracle query to L checking if $H^i \in L$ and if all answers are no, reject otherwise accept.

We now argue the correctness of the reduction. Suppose $det(M)$ is 0, then $N = g(M)$ (by Fact 5.10.2) must have full rank. Hence for all $1 \leq i \leq d$, $\mathsf{rowspan}(M_1^i)$ and $\mathsf{rowspan}(M_2^i)$ do not intersect. If $det(M) \neq 0$, then $N = g(M)$ (by Fact 5.10.2) must have a linearly dependent column and hence there is some i for which $\mathsf{rowspan}(M_1^i)$ and $\mathsf{rowspan}(M_2^i)$ is non-zero. Also the overall reduction runs in logspace as g is logspace computable.

The upper bound for L follows by observing that given two $d \times d$ matrices M_1 and M_2, their individual ranks r_1 and r_2 can be computed in $\mathsf{L}^{\mathsf{C}_=\mathsf{L}}$ [ABO99]. Consider the matrix M of size $d \times 2d$ by adjoining M_1 and M_2. It follows that the $rowspace(M_1) \cap rowspace(M_2) \neq \phi$ if and only if $\mathsf{rank}(M) < r_1 + r_2$. The latter condition can also be tested using a query to a $\mathsf{C}_=\mathsf{L}$ oracle. □

5.11 Other variants of projective dimension and their connection to graph theoretic parameters

In this section, we study two stringent variants of projective dimension for which exponential lower bounds and exact characterizations can be derived. Although these measure do not correspond to restrictions on branching programs, they illuminate the essential nature of the general measure. We define the measures and show their characterizations in terms of well-studied graph theoretic parameters.

Definition 5.11.1 (Standard Projective Dimension). A Boolean function $f : \{0,1\}^n \times$

$\{0,1\}^n \to \{0,1\}$ with the corresponding bipartite graph $G(U,V,E)$ is said to have standard projective dimension (denoted by $\mathsf{spd}(f)$) d over field $\mathbb{F}$, if d is the smallest possible dimension for which there exists a vector space K of dimension d over $\mathbb{F}$ with a map ϕ assigning subspaces of K to $U \cup V$ such that

- for all $(u,v) \in U \times V$, $\phi(u) \cap \phi(v) \neq \{0\}$ if and only if $(u,v) \in E$.
- $u \in U \cup V$, $\phi(u)$ is spanned by a subset of standard basis vectors in K.

In addition to the above constraints, if the assignment satisfies the property that for all $(u,v) \in U \times V$, $\mathsf{dim}\left(\phi(u) \cap \phi(v)\right) \leq 1$, we say that the *standard projective dimension is with intersection dimension* 1, denoted by $\mathsf{uspd}(f)$. We make some easy observations about the definition itself.

For $N \times N$ bipartite graph G with m edges, consider the assignment of standard basis vectors to each of the edges and for any $u \in U \cup V$, $\phi(u)$ is the span of the basis vectors assigned to the edges incident on u. Moreover, the intersection dimension in this case is 1. Hence for any G, $\mathsf{spd}(G) \leq \mathsf{uspd}(G) \leq m$.

Even though $\mathsf{pd}(G) \leq \mathsf{spd}(G)$, there are graphs for which the gap is exponential. For example, consider the bipartite realization G of EQ_n with $N = 2^n$. We know $\mathsf{pd}(G) = \Theta(\log N)$ but $\mathsf{spd}(G) \geq N$ since each of the vertices associated with the matched edges cannot share any basis vector with vertices in other matched edges. Hence dimension must be at least N. We show that standard projective dimension of a bipartite G is the same as its biclique cover number.

Definition 5.11.2 (Biclique cover number)**.** For a graph G, a collection of complete bipartite graphs defined on $V(G)$ is said to cover G if every edge in G is present in some complete bipartite graph of the collection. The size of the smallest collection of bipartite graphs which covers G is its biclique cover number (denoted by $bc(G)$).

If in addition, we insist that bicliques must be edge-disjoint, the parameter is known as *biclique partition number* denoted by $bp(G)$.

Theorem 5.11.3 (Restatement of Theorem 5.1.4)**.** *For any Boolean function f, $bc(G_f) = \mathsf{spd}(G_f)$ and $\mathsf{uspd}(G_f) = bp(G_f)$.*

Proof. ($\mathsf{spd}(f) \leq bc(G_f)$) Let $G = G_f$, $t = bc(G)$ and $A_1 \ldots, A_t$ be a bipartite cover for G. For a vertex $v \in V(G)$, let $I_v = \{e_i \mid v \in A_i\}$. We claim that $\{I_v\}_{v \in V(G)}$ is a valid standard projective assignment. Suppose $I_u \cap I_v \neq \emptyset$, then there exists an i such that $u, v \in A_i$ and $(u, v) \in E(A_i)$. Hence $(u, v) \in E(G)$. Also if $(u, v) \in E(G)$, then $\exists\ i$ s.t. $(u, v) \in E(A_i)$. By definition of I_u, I_v, $e_i \in I_u \cap I_v$ giving $I_u \cap I_v \neq \emptyset$.

($bc(G_f) \leq \mathsf{spd}(G_f)$) Let $G = G_f, t = \mathsf{spd}(G)$ and $\{I_u\}_{u \in V(G)}$ be the subsets assigned. Consider $G_i = \{(u, v) \mid i \in I_u \text{ and } i \in I_v\}$ for $i \in \{1, \ldots, t\}$. We claim that the collection of G_i forms a valid bipartite cover of G. If $(u, v) \in E(G)$, we have $I_u \cap I_v \neq \emptyset$. Hence there exists an $i \in I_u \cap I_v$ and $(u, v) \in E(G_i)$. If $(u, v) \in E(G_i)$ for some i, then $i \in I_u$ and $i \in I_v$ implying $I_u \cap I_v \neq \emptyset$. This gives that $(u, v) \in E(G)$ from the definition of standard assignment.

($bp(G_f) \leq \mathsf{uspd}(G_f)$) Let ϕ be the intersection dimension one standard assignment of ambient dimension d of f. For every $e_i \in \mathbb{F}^d$, define the set $C_i = \left\{(x, y) \mid \phi(x, y) = e_i\right\}$. We claim that $\mathcal{C} = \{C_i\}_{i \in [d]}$ is a bipartite partition of G_f. Every C_i thus defined is a biclique, because if $\phi(x, y) = e_i$ then that implies $e_i \in \phi(x)$ and $e_i \in \phi(y)$. Note that for every $(x, y) \in G_f$, there exists a unique $i \in [d]$ such that $\phi(x, y) = e_i$. Hence any $(x, y) \in G_f$ belongs to exactly one of the sets C_i thus implying that C_i's are edge disjoint biclique covers. Note that any $(x, y) \notin G_f$ do not belong to any of C_i's as $\phi(x, y) = \{0\}$.

($\mathsf{uspd}(G_f) \leq bp(G_f)$) Let $\mathcal{C} = \{C_i\}_{i \in [d]}$ where $d = bp(G_f)$ be a biclique partition

cover. We give a standard assignment ϕ for G_f defined as follows. For any x, $\phi(x) = \mathsf{span}\{e_i \mid \exists y, (x,y) \in C_i\}$. By definition ϕ is a standard assignment. We just need to prove that $(x,y) \in G_f$ if and only $\phi(x,y) \neq \{0\}$ and $\mathsf{dim}\phi(x,y) = 1$. To prove this we would once again employ the rectangle property of bicliques, that is if $(x,y') \in C_i$ and $(x',y) \in C_i$ then so is (x,y). First we will argue that if there an intersection then it is dimension 1. Recall that intersection of two standard subspaces is a standard subspace. Suppose there exists (x,y) with $\mathsf{dim}\phi(x,y) > 1$. Let e_j, e_k be any two standard intersection vectors in $\phi(x,y)$. By construction and the rectangle property of bicliques, we get that $(x,y) \in C_j$ and $(x,y) \in C_k$ contradicting the disjoint cover property. Hence for any (x,y), $\mathsf{dim}\phi(x,y) \leq 1$. If $(x,y) \notin G_f$, then there does not exist an i, $(x,y) \in C_i$. But if $\phi(x,y) = e_i$ for some $i \in [d]$, then that implies by the rectangle property of bicliques that $(x,y) \in C_i$, a contradiction. □

5.12 Discussion - ”natural”-ness of projective dimension

We conclude this chapter with a discussion on the naturalness of this approach. Since even circuit classes like NC^1 and TC^0 are believed to contain pseudo-random functions, the natural proofs barrier applies to class L which contains all these classes.

Recall that one of the properties a natural proof need to satisfy is the “constructiveness”. Constructiveness of a property $\mathcal{P}$, says that given a truth table of a Boolean function f, it can be decided whether the function satisfies the property or not, in polynomial (in the size of the truth table) time. In case of a projective dimension based super polynomial lower bound, the property under consideration

would be, given a Boolean function f is it true that $\mathsf{pd}(f) = n^{\omega(1)}$? We do not even know how to check even for a weaker property like, is it true that $\mathsf{pd}(f) = \Omega(n^2)$. A naive algorithm to do this would have to try all possible linear sub-spaces of $\mathbb{F}_2^{n^2}$ for all vertices of G_f, the natural bipartite graph associated with f. Though number of vertices in G_f is bounded by truth table size, this kind of brute-force enumeration, trying to find the correct subspace for each vertex, takes exponential time in the number of vertices and logarithm in the number of linear subspaces of $\mathbb{F}_2^{n^2}$. The number of subspace of $\mathbb{F}_2^{n^2}$ is already exponential in n, hence the logarithm of this quantity is polynomial in n. Thus, this naive strategy is exponential in truth table size. Thus, we believe that the projective dimension based approach is non-natural.

Regarding bitpdim, since we have proved that it is equivalent to branching program size up to polynomial factors it is not hard to see that constructiveness also doesn't hold. This is because, the property that a Boolean function has bitpdim $\mathsf{bitpdim}(f) = n^{\omega(1)}$ is equivalent to the function not having polynomial sized branching programs. There are no algorithms known to check if a function has no polynomial sized branching programs, which does better than the naive algorithm, which is to enumerate all polynomial sized branching programs and check if the compute f one by one. This naive algorithm is not poly in the size of the truth table of f. Hence we believe that bitpdim based approach is also non-natural.

CHAPTER 6

Discussions and Open problems

We studied lower bounds against non-monotone circuits with a new measure of non-monotonicity - namely the orientation of the functions computed at each gate of the circuit. As the first step, we proved that the lower bound can be obtained by modifying the Karchmer–Wigderson game. We studied the weight of the orientation of the functions at internal gates as a parameter of the circuit, and explored the usefulness of densely oriented gates. We also showed the connections between negation limited circuits and orientation limited circuits. A main open problem that arises from our work is to improve upon the weight restriction of the orientation vector ($\Omega(\frac{\sqrt{N}}{\log N})$) for which we can prove depth lower bounds.

Based on the techniques employed in the lower bound of [BCO+15a], the hardness amplification of learning composition of functions based on a Fourier analytic property Expected Bias ([FLS11], [O'D04]), we come up with information theoretic noise model tailored for sparse orientation. We also outline a strategy which our strategy can be executed to get a learning lower bound for Boolean functions whose weight of orientation is limited, assuming Conjecture 4.3.8, provided one can analyze noise stability of Boolean functions under the noise model where are two noise operators operating on two parts of the input to the function. The two open problems listed below, we believe, are important questions not just from the point of the view of the lower bound, but are also interesting from the point of view of Fourier analysis of Boolean functions. The first problem is to analyze the expected bias of REC-3-MAJ : $\{0,1\}^k \to \{0,1\}$ under the noise model we defined.

That is analyze,

$$\text{ExpBias}_{\gamma(N_1,\delta_1,\delta_2)}(\text{REC-3-MAJ}) = \mathsf{E}_{\rho \in P^k_{N_1,\delta_1,\delta_2}} \left[\text{bias}(\text{REC-3-MAJ}_\rho)\right]$$

where $\rho \in P^k_{N_1,\delta_1,\delta_2}$ is defined to be,

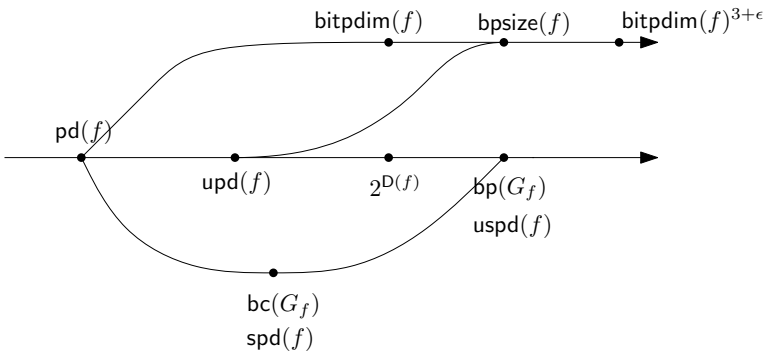
$$\rho(i) \quad = \quad \begin{cases} \star & \text{with probability } 2\delta_1, i \in N_1 \\ 0 & \text{with probability } \frac{1-2\delta_1}{2}, i \in N_1 \\ 1 & \text{with probability } \frac{1-2\delta_2}{2}, i \in N_1 \\ \star & \text{with probability } 2\delta_2, i \in [k] \setminus N_1 \\ 0 & \text{with probability } \frac{1-2\delta_2}{2}, i \in [k] \setminus N_1 \\ 1 & \text{with probability } \frac{1-2\delta_2}{2}, i \in [k] \setminus N_1 \end{cases}$$

The second problem is to analyze the expected bias of the Mossel O'Donnell function [MO03] composed with majority. For $c = 3$, $N_1 = \{(i,1),(i,2)\}_{i\in[r]}$, analyze the expected bias of $f_3 : \{0,1\}^{3r} \to \{0,1\}$ under the noise model we defined. That is analyze,

$$\text{ExpBias}_{\gamma(N_1,\delta_1,\delta_2)}(f_3) = \mathsf{E}_{\rho \in P^k_{N_1,\delta_1,\delta_2}} \left[\text{bias}(f_{3_\rho})\right]$$

We studied variants of projective dimension of graphs with improved connection to branching programs. We showed lower bounds for these measures indicating the weakness and of each of the variants. A pictorial representation of all parameters is shown in Fig. 6.1.

An immediate question that arises from our work is whether $\Omega(d^2)$ lower bound on $\mathsf{upd}(\mathcal{P}_d)$ is tight. In this direction, since we have established a gap between $\mathsf{upd}(\mathcal{P}_d)$ and $\mathsf{pd}(\mathcal{P}_d)$, it is natural to study how pd and upd behave under composi-



D(f) – Deterministic Communication Complexity of f
bc(G) – Bipartite Cover number of G
bp(G) – Bipartite Partition number of G

Figure 6.1: Parameters considered in this work and their relations

tion of functions, in order to amplify this gap.

In another direction, we believe that the $\Omega(d^2)$ lower bound on upd$(\mathcal{P}_d)$ is not tight. It is natural to study composition of functions to improve this gap.

The subspace counting based lower bounds for bitpdim that we proved are tight for functions like ED_n. However, observe that under standard complexity theoretic assumptions the bitpdim assignment for $\mathcal{P}_d$ is not tight. Hence it might be possible to use the specific linear algebraic properties of $\mathcal{P}_d$ to improve the bitpdim lower bound we obtained for $\mathcal{P}_d$.

Publications

Conference Publications

1. Depth lower bounds against circuits with sparse orientation.
   Sajin Koroth, **Jayalal Sarma**
   Depth lower bounds against circuits with sparse orientation. In Zhipeng Cai, Alex Zelikovsky, and Anu G. Bourgeois, editors, *Computing and Combinatorics - 20th International Conference, COCOON 2014, Atlanta, GA, USA, August 4-6, 2014. Proceedings*, volume 8591 of *Lecture Notes in Computer Science*, pages 596–607. Springer, 2014. An extended version to appear in volume 152 of Fundamenta Informaticae.

2. Characterization and Lower Bounds for Branching Program Size Using Projective Dimension
   Krishnamoorthy Dinesh, **Sajin Koroth**, **Jayalal Sarma**
   In Akash Lal, S. Akshay, Saket Saurabh, and Sandeep Sen, editors, *36th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS 2016, December 13-15, 2016, Chennai, India*, volume 65 of *LIPIcs*, pages 37:1–37:14. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2016.

Journal Publications

1. Depth Lower Bounds against Circuits with Sparse Orientation
   Sajin Koroth, **Jayalal Sarma**
   The paper includes and extends the results presented in the conference version listed above. Accepted to the journal *Fundamenta Informaticae*, Vol 152, 2017

Journal Submissions under Review

1. Characterization and Lower Bounds for Branching Program Size Using Projective Dimension
   Krishnamoorthy Dinesh, **Sajin Koroth**, **Jayalal Sarma**
   The paper includes and extends the results presented in the conference ver-

sion listed above. *Journal version submitted to ACM Transactions on Computation Theory (TOCT)* in Jan 2017. Under review.

REFERENCES

[Aar17] Scott Aaronson. P=?NP. *Electronic Colloquium on Computational Complexity (ECCC)*, 24:4, 2017.

[AB87] Noga Alon and Ravi B. Boppana. The Monotone Circuit Complexity of Boolean Functions. *Combinatorica*, 7(1):1–22, 1987.

[AB07] S. Arora and B. Barak. *Computational Complexity A Modern Approach*. Cambridge University Press, 2007.

[ABO99] E. Allender, R. Beals, and M. Ogihara. The complexity of matrix rank and feasible systems of linear equations. *Computational Complexity*, 8(2):99–126, 1999.

[AKL+79] Romas Aleliunas, Richard M. Karp, Richard J. Lipton, László Lovász, and Charles Rackoff. Random walks, universal traversal sequences, and the complexity of maze problems. In *20th Annual Symposium on Foundations of Computer Science, San Juan, Puerto Rico, 29-31 October 1979*, pages 218–223, 1979.

[AM96] K. Amano and A. Maruoka. Potential of the approximation method. In *Proceedings of 37th Annual Symposium on Foundations of Computer Science*, pages 431 –440, oct 1996.

[AM05] Kazuyuki Amano and Akira Maruoka. A superpolynomial lower bound for a circuit computing the clique function with at most (1/6) log log n negation gates. *SIAM Journal on Computing*, 35(1):201–216, 2005.

[BCO+15a] Eric Blais, Clément L. Canonne, Igor Carboni Oliveira, Rocco A. Servedio, and Li-Yang Tan. Learning circuits with few negations. In *APPROX-RANDOM*, volume 40 of *LIPIcs*, pages 512–527. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2015.

[BCO+15b] Eric Blais, Clément L. Canonne, Igor Carboni Oliveira, Rocco A. Servedio, and Li-Yang Tan. Learning circuits with few negations. In Naveen Garg, Klaus Jansen, Anup Rao, and José D. P. Rolim, editors, *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2015, August 24-26, 2015, Princeton, NJ, USA*, volume 40 of *LIPIcs*, pages 512–527. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2015.

[BGMS16] Paul Beame, Nathan Grosshans, Pierre McKenzie, and Luc Segoufin. Nondeterminism and an abstract formulation of nečiporuk's lower bound method. *CoRR*, abs/1608.01932, 2016.

[BM06] D.C. Brock and G.E. Moore. *Understanding Moore's Law: Four Decades of Innovation*. Chemical Heritage Foundation, 2006.

[Bol01] Béla Bollobás. *Random Graphs, Second edition*. Cambridge Studies in Advanced Mathematics 73. Cambridge University Press, 2001.

[BT96] Nader H Bshouty and Christino Tamon. On the fourier spectrum of monotone functions. *Journal of the ACM (JACM)*, 43(4):747–770, 1996.

[Del76] Philippe Delsarte. Association schemes and t-designs in regular semilattices. *Journal of Combinatorial Theory, Series A*, 20(2):230–243, mar 1976.

[DKS16] Krishnamoorthy Dinesh, Sajin Koroth, and Jayalal Sarma. Characterization and lower bounds for branching programs size using projective dimension. In *The 36th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS 2016, Chennai, India, December 13-15, 2016*, 2016.

[Edm65] Jack Edmonds. Paths, trees, and flowers. *Canad. J. Math.*, 17:449–467, 1965.

[ER60] P. Erdös and R. Rado. Intersection theorems for systems of sets. *Journal of the London Mathematical Society*, s1-35(1):85–90, 1960.

[FG85] Péter Frankl and Ronald L Graham. Intersection theorems for vector spaces. *European Journal of Combinatorics*, 6(2):183–187, jun 1985.

[Fis75] Michael Fischer. The Complexity of Negation-limited Networks — a Brief Survey. In *Automata Theory and Formal Languages 2nd GI Conference Kaiserslautern*, volume 33 of *Lecture Notes in Computer Science*, pages 71–82. 1975.

[FLS11] Vitaly Feldman, Homin K. Lee, and Rocco A. Servedio. Lower bounds and hardness amplification for learning shallow monotone formulas. In Sham M. Kakade and Ulrike von Luxburg, editors, *COLT 2011 - The 24th Annual Conference on Learning Theory, June 9-11, 2011, Budapest, Hungary*, volume 19 of *JMLR Proceedings*, pages 273–292. JMLR.org, 2011.

[FSS84] Merrick Furst, James B. Saxe, and Michael Sipser. Parity, circuits, and the polynomial-time hierarchy. *Mathematical systems theory*, 17(1):13–27, 1984.

[FW86] Péter Frankl and Richard M Wilson. The Erdős-Ko-Rado theorem for vector spaces. *Journal of Combinatorial Theory Series A*, 43(2):228–236, nov 1986.

[Hal74] P.R. Halmos. *Finite-Dimensional Vector Spaces*. Undergraduate Texts in Mathematics. Springer, 1974.

[Imp95] Russell Impagliazzo. Hard-core distributions for somewhat hard problems. In *In 36th Annual Symposium on Foundations of Computer Science*, pages 538–545. IEEE, 1995.

[IPS97] Russell Impagliazzo, Ramamohan Paturi, and Michael E. Saks. Size-depth tradeoffs for threshold circuits. *SIAM Journal of Computing*, 26(3):693–707, 1997.

[JÓ2] Joseph JáJá. *An Introduction to Parallel Algorithms*. Addison Wesley Longman Publishing Co., Inc., Redwood City, CA, USA, 1992.

[Juk12] Stasys Jukna. *Boolean Function Complexity: Advances and Frontiers*, volume 27 of *Series: Algorithms and Combinatorics*. Springer New York Inc., 2012.

[KKS16a] Dinesh Krishnamoorthy, Sajin Koroth, and Jayalal Sarma. Characterization and lower bounds for branching program size using projective dimension. In Akash Lal, S. Akshay, Saket Saurabh, and Sandeep Sen, editors, *36th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS 2016, December 13-15, 2016, Chennai, India*, volume 65 of *LIPIcs*, pages 37:1–37:14. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2016.

[KKS16b] Dinesh Krishnamoorthy, Sajin Koroth, and Jayalal Sarma. Characterization and lower bounds for branching program size using projective dimension. *Electronic Colloquium on Computational Complexity (ECCC)*, 23:76, 2016.

[KL82] R M Karp and R Lipton. Turing machines that take advice. *L'Enseignement Mathematique*, (28), 1982.

[KN97] Eyal Kushilevitz and Noam Nisan. *Communication Complexity*. Cambridge University Press, New York, NY, USA, 1997.

[Kor17] Sajin Koroth. Learning algorithms for circuits of sparse orientation. Manuscript available on request, 2017.

[KS14] Sajin Koroth and Jayalal Sarma. Depth lower bounds against circuits with sparse orientation. In Zhipeng Cai, Alex Zelikovsky, and Anu G. Bourgeois, editors, *Computing and Combinatorics - 20th International Conference, COCOON 2014, Atlanta, GA, USA, August 4-6, 2014. Proceedings*, volume 8591 of *Lecture Notes in Computer Science*, pages 596–607. Springer, 2014. An extended version to appear in volume 152 of Fundamenta Informaticae.

[KS17] Sajin Koroth and Jayalal Sarma. Depth lower bounds against circuits with sparse orientation. *Fundamenta Informaticae*, 152, 2017.

[KW88] Mauricio Karchmer and Avi Wigderson. Monotone circuits for connectivity require super-logarithmic depth. In Janos Simon, editor, *Proceedings of the 20th Annual ACM Symposium on Theory of Computing, May 2-4, 1988, Chicago, Illinois, USA*, pages 539–550. ACM, 1988.

[KW90] M. Karchmer and A. Wigderson. Monotone circuits for connectivity require super-logarithmic depth. *SIAM Journal on Discrete Mathematics*, 3(2):255–265, 1990.

[Lov79] László Lovász. On determinants, matchings, and random algorithms. In *Symposium on Fundamentals of Computation Theory (FCT)*, pages 565–574, 1979.

[LW12] Benjian Lv and Kaishun Wang. The eigenvalues of q-Kneser graphs. *Discrete Mathematics*, 312(6):1144 – 1147, 2012.

[Mar58] A. A. Markov. On the inversion complexity of a system of functions. *J. ACM*, 5(4):331–334, October 1958.

[MO03] Elchanan Mossel and Ryan O'Donnell. On the noise sensitivity of monotone functions. *Random Structures and Algorithms*, 23(3):333–350, 2003.

[MU05] Michael Mitzenmacher and Eli Upfal. *Probability and Computing: Randomized Algorithms and Probabilistic Analysis*. Cambridge University Press, New York, NY, USA, 2005.

[Nec66] E. I. Nechiporuk. On a boolean function. *Doklady of the Academy of Sciences of the USSR*, 164(4):765–766, 1966.

[O'D04] Ryan O'Donnell. Hardness amplification within {NP}. *Journal of Computer and System Sciences*, 69(1):68 – 94, 2004. Special Issue on Computational Complexity 2002.

[PR92] P. Pudlák and V. Rödl. A combinatorial approach to complexity. *Combinatorica*, 12:221–226, 1992.

[PR94] P. Pudlák and V. Rödl. Some combinatorial-algebraic problems from complexity theory. *Discrete Mathematics*, 136(1-3):253–279, dec 1994.

[Raz85a] AA Razborov. Lower Bounds for Monotone Complexity of Some Boolean Functions. *Soviet Math. Doklady.*, pages 354–357, 1985.

[Raz85b] Alexander A Razborov. Lower bounds on monotone complexity of the logical permanent. *Mathematical Notes*, 37(6):485–493, 1985.

[Raz87] A. A. Razborov. Lower bounds on the size of bounded depth circuits over a complete basis with logical addition. *Mathematical notes of the Academy of Sciences of the USSR*, 41(4):333–338, 1987.

[RBG02] Lajos Rónyai, László Babai, and Murali K. Ganapathy. On the number of zero-patterns of a sequence of polynomials. *Journal of the AMS*, 14:2001, 2002.

[Rei08] Omer Reingold. Undirected connectivity in log-space. *Journal of the ACM*, 55(4):17:1–17:24, September 2008.

[Rom05] Steven Roman. *Advanced Linear Algebra*, volume 135 of *Graduate Texts in Mathematics*. Springer Science & Business Media, 2005.

[RR97] Alexander A Razborov and Steven Rudich. Natural proofs. *Journal of Computer and System Sciences*, 55(1):24 – 35, 1997.

[RSA78] R. L. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Commun. ACM*, 21(2):120–126, February 1978.

[RW89] Ran Raz and Avi Wigderson. Probabilistic communication complexity of boolean relations. In *Proc. of the 30th FOCS*, pages 562–567, 1989.

[RW92] Ran Raz and Avi Wigderson. Monotone circuits for matching require linear depth. *Journal of ACM*, 39(3):736–744, July 1992.

[Smo87] R. Smolensky. Algebraic methods in the theory of lower bounds for boolean circuit complexity. In *Proceedings of the Nineteenth Annual ACM Symposium on Theory of Computing*, STOC '87, pages 77–82, New York, NY, USA, 1987. ACM.

[Tal96] Michel Talagrand. How much are increasing sets positively correlated? *Combinatorica*, 16(2):243–258, 1996.

[Tar88] Éva Tardos. The gap between monotone and non-monotone circuit complexity is exponential. *Combinatorica*, 8(1):141–142, 1988.

[Tur36] Alan Turing. On computable numbers with an application to the " Entscheidungsproblem". *Proceeding of the London Mathematical Society*, 1936.

[Vol99] Heribert Vollmer. *Introduction to Circuit Complexity: A Uniform Approach*. Springer New York Inc., 1999.

CURRICULUM VITAE

1. NAME : Sajin Koroth

2. DATE OF BIRTH : 5th April, 1987

3. EDUCATIONAL QUALIFICATIONS

Bachelor of Technology

Institute	:	Government College of Engineering, Kannur
Major	:	Computer Science and Engineering
Year	:	July 2009 - July 2011

M.S. by Research

Institute	:	Indian Institute of Technology, Madras
Research Area	:	Combinatorics
Year	:	July 2009 - July 2011

DOCTORAL COMMITTEE

CHAIRPERSON : Prof. C. Pandu Rangan
Professor
Department of Computer Science and Engineering

GUIDE : Dr. Jayalal Sarma
Associate Professor
Department of Computer Science and Engineering

MEMBER : Dr. N.S. Narayanaswamy
Professor
Department of Computer Science and Engineering

MEMBER : Dr. Raghavendra Rao B. V.
Assistant Professor
Department of Computer Science and Engineering

Dr. Andrew Thangaraj
Professor
Department of Electrical Engineering

MEMBER : Narayanan N
Assistant Professor
Department of Mathematics