Exploring Different Models of Query Complexity and Communication Complexity

Supartha Podder

(M.Sc.(CS), École normale supérieure de Cachan)

A thesis submitted in fulfilment of the requirements

for the degree of Doctor of Philosophy (Ph.D.)

in the

Centre for Quantum Technologies,

National University of Singapore,

Singapore

Centre for Quantum Technologies

NUS National University of Singapore

2016

DECLARATION

I hereby declare that this thesis is my original work and it has been written by me in its entirety.

I have duly acknowledged all the sources of information which have been used in the thesis.

This thesis has also not been submitted for any degree in any university previously.

Supartha Podder

16 September 2016

ACKNOWLEDGEMENTS

There are many wonderful people who, in a direct or indirect way, helped me making this report that you are either reading on your electronic screen or holding in your hand. I would like to thank all of them.

First and foremost I am deeply indebted to my supervisor Hartmut Klauck for giving me the opportunity to work with him and for giving me his guidance and support throughout my graduate career. I am grateful to him for his valuable advice and feedback regarding all aspects of my Ph.D. work from doing research, writing papers to applying for postdocs. I am greatly benefited from the various discussions that we had over the years. I also thank him for supporting me during the difficult phases of my Ph.D. life.

I would like to thank all my co-authors (Hartmut Klauck, Raghav Kulkarni, Samir Datta, Nikhil Balaji) for many enjoyable brainstorming sessions and fruitful research discussions.

My sincere thanks to Miklos Santha, Rahul Jain, Troy Lee, Dmitry Gavinsky, Iordanis Kerenidis and Gabor Ivanyos for many helpful discussions over the years.

I would like to express my gratitude to friends and colleagues at CQT and elsewhere for their numerous support and discussions. A partial list includes – Raghav Kulkarni, Tanvirul Islam, Attila Pereszlényi, Anurag Anshu, Aarthi Sundaram, Priyanka Mukhopadhyay, Maharshi Ray, Jamie Sikora, Antonios Varvitsiotis, Laura Mančinska, Sambit Bikas Pal, Ved Prakash, Penghui Yao, Wei Zhaohui, Sarvagya Upadhyay, and Thomas Decker. I am also thankful to all the administrators of CQT, specially Evon Tan and Chan Chui Theng, for their excellent administrative support. CQT also deserves special mention for providing me scholarship, a nice office space, friendly research environment and a free coffee machine.

My humble salutation to my previous mentors and advisors, Iordanis Kerenidis, Meena Mahajan, Saptarshi Naskar and Soumen Saha. Because of their motivation and guidance, I could pursue computer science.

I am very grateful to my father, Bishnu Podder, for his constant encouragement and supporting me in whatever I do. My humble salutations to him, my mother Sudipta Podder and my brother Partha Protim Poddar. Lastly I would like to thank my wife Jaya Roy for her countless love, support and advice. She made a lot of sacrifices for me. I dedicate this thesis to my father and my wife.

Contents

Summary

This thesis has two parts. In the first part, we study graph properties in the query complexity model. In particular first we focus on the subgraph isomorphism and subgraph homomorphism problem. The subgraph isomorphism problem is a generalization of the well-known graph isomorphism problem, where one asks whether a given graph H is isomorphic to a subgraph of another graph G. We give new lower bounds for the quantum query complexity of these problems.

The study of the subgraph isomorphism problem motivates the study of a new node-query setting in the query complexity of graph properties. In chapter 6, we introduce the new node query setting and study several graph properties in that setting. In particular, we focus on the effect of removing symmetry from graphs and study how low the query complexity can fall if we take away the symmetry assumption from the input graph. By taking away the symmetry we mean, first we ask what is the minimum cost of finding a property in graphs when the graphs have a certain amount of symmetry. Then we compare it with the cost of finding the property in graphs which have no such restrictions.

The second part is about communication complexity and communication protocols. In this part we try to understand the power of quantum proofs and quantum messages in the communication complexity setting. We study whether having quantum resources gives us any advantage over having classical resources. As we shall see in certain scenarios the answer is yes, whereas in some other scenarios the answer is no. We also investigate how computation is affected if we restrict ourself to a certain class of communication protocols – in particular we focus on the garden-hose game, which is a memoryless and reversible communication model.

Structure of the Thesis

Below we summarize the results of all the chapters in this thesis.

- The first three chapters are introductory chapters and contain preliminaries needed for the rest of the thesis. In Chapter 2, we review some relevant aspects of quantum computing and quantum information which are essential for this thesis. In Section 2.2 of Chapter 2, the concepts of basic linear algebra, Hilbert space, quantum states, and different quantum operators are introduced. Section 2.2.6 reviews how to represent quantum states as density operators. Finally in Section 2.3, we explore some quantum information measures and properties which will be useful later on.

- In Chapter 3, we introduce query complexity and review different models of it. In Section 3.3, we explore various query complexity measures which are used to give lower bounds to different models of query complexity. In Section 3.4, we summarize the relationship between different complexity measures and query complexity models.

- Chapter 4 introduces different versions of the communication complexity model. In Section 4.3, we define Merlin-Arthur protocols, which are the communication complexity analogue of the Turing machine class MA. For the purpose of this thesis we focus on one-way Merlin Arthur protocols.

- In Chapter 5, we study the query complexity of the subgraph isomorphism problem and subgraph homomorphism problem. One of the most intriguing problems in computer science is the Graph Isomorphism Problem: determining whether two finite graphs are isomorphic. The graph isomorphism problem is not known to be solvable in polynomial time. It is also highly unlikely that it will be a NP-complete problem. Thus the problem may be in the complexity class NP-intermediate. Very recently a quasi-polynomial time algorithm for Graph Isomorphism problem has been proposed by Babai [Bab15]. The Subgraph Isomorphism Problem, on the other hand, denoted by f_H, is a generalization of the Graph Isomorphism Problem where one asks whether a graph H is isomorphic to a *subgraph* of another graph G. Unlike the graph isomorphism problem this problem is NP-Complete. Several central computational problems for graphs such as containing a clique, containing a Hamiltonian cycle, containing a perfect matching can be formulated as instances of the

Subgraph Isomorphism Problem by fixing the H appropriately. We investigate this problem in the quantum setting and among other results we obtain a quantum query complexity lower bound for f_H, i.e., $Q(f_H) = \Omega(n^{3/4})$. Previously a trivial lower bound of $\Omega(\sqrt{n})$ was known for this problem. In Section 5.3, we study the Subgraph Homomorphism Problem, denoted by $f_{[H]}$, and show that $Q(f_{[H]}) = \Omega(n)$. We also extend both our results to the 3-uniform hypergraphs.

- Chapter 6 is a continuation of the study of graph properties in query complexity. The query complexity of graph properties is well-studied when the queries are on the edges. We investigate the same when the queries are on the nodes. In this setting, given a fixed and known graph the nodes of the graph can be either active or inactive. The set of all active nodes induces a sub-graph and we can ask whether the induced sub-graph satisfies a certain property. This model turns out to be combinatorially rich. Our main motivation to study this model comes from the fact that it allows us to initiate a systematic study of breaking symmetry in the context of query complexity of graph properties. It turns out that even with a minimal symmetry assumption on G, namely that of vertex-transitivity, the query complexity for any hereditary graph property in this setting is the worst possible, i.e., n. We show that in the absence of any symmetry on G every hereditary property benefits at least quadratically. Furthermore the query complexity cannot fall exponentially low for any hereditary property. We study several other important graph properties in this setting.

- From Chapter 7 onwards we focus on communication complexity and communication protocols. We investigate the relationship between quantum and classical messages. In Section 7.2, we show how to replace a quantum message in a one-way communication protocol by a deterministic message, establishing that for all partial Boolean functions $f : \{0,1\}^n \times \{0,1\}^m \to \{0,1\}$ we have $D^{A\to B}(f) \leq O(Q^{A\to B,*}(f) \cdot m)$. We also give the first example of a scenario in which for quantum verifiers quantum proofs lead to exponential savings in computing a classical Boolean function compared to quantum verifiers using classical proofs. To do this we exhibit a partial Boolean function f for which a QMA protocol has cost $O(\log n)$, whereas every QCMA protocol for f requires $\Omega(\sqrt{n}/\log n)$ communication (in the one-way setting).

- In Chapter 8 we study a restricted distributed communication model called the garden-hose model. Chapter 8 contains new results about the garden-hose model. In this chapter

we investigate how computation is affected if we restrict ourselves to a certain class of communication protocols – in particular we focus on the recently proposed garden-hose model [BFSS13], which is a memoryless and reversible communication model. We shall see that this model has beautiful connections with several other computational models and can be used to derive new lower bounds for other computational models. Investigation of this model will also broaden our understanding of time and space complexity and their trade-offs.

- Chapter 9 concludes this thesis and discusses some related open problems.

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Motivation

One of the most important and central topics in theoretical computer science is to understand whether some problems are harder than others in terms of the amount of computational resources needed to solve these problems. For instance, we still don't know if addition is inherently easier than multiplication in the Turing machine model. And if yes then why? Currently addition has a linear time algorithm, but the best algorithm for the integer multiplication, due to De et al. [DKSS13], works in $O(n \log n \log \log n)$ time. Another intriguing problem is the graph isomorphism problem. The problem is neither known to be solvable in polynomial time by a Turing machine nor it is known to be NP-complete. Yet another example is the factorization problem: when using quantum resources it has a polynomial time algorithm, but it is not known to have a polynomial time algorithm otherwise. These problems, with the many other such problems, compel us to understand what is it that makes these problems so different from each other. In general the broader goal of complexity theory is to classify problems according to these inherent difficulties and resources (e.g. time, space etc.) needed.

Very often the underlying model of such investigations is the Turing machine model. Turing machines have a simplistic yet powerful design and the famous Church-Turing thesis hypothesizes that any physically realizable computing device can be simulated by a universal Turing machine. Thus Turing machines are at the centre of complexity theory research. A recurrent theme in the literature is to investigate how computation is affected by the presence or absence of different resources and to explore the connections between them. One such problem is the biggest open problem in computer science, namely the P vs. NP problem, which asks whether non-determinism

gives any advantage for polynomial time Turing machines. Another important open problem is the P vs. BPP problem, which asks whether or not randomization helps polynomial time Turing machines.

Unfortunately it is very difficult to prove good lower bounds for the Turing machine model and despite much effort, there is no significant lower bound that could be proven. Thus it is worth looking into simpler models which are easier to analyse and in which it is easier to prove lower bounds. But the models should be worthy of investigation i.e., studying these models should shed some more light into why some problems are inherent harder than the others and should broaden our understanding of the different computational resources, and computation in general.

The query complexity and the communication complexity model perfectly fit into the picture. Here in these models we can actually prove some lower bounds and exhibit the difference between different computational resources. Query complexity is the study of how many input bits of an n-bit input x we need to read in order to find the output of a function f on input x, $f(x)$. For example, consider the AND-function, which is 1 iff all the input bits are 1. Now in evaluating the AND-function, if we find a single input bit that is 0, we can immediately stop and output 0. We don't have to look at the remaining bits. But in the worst case, if the input consists of all 1s, we would have to look at all of the input positions before we can be sure that the output is indeed 1. Because if we only check $n - 1$ bits and produce the answer, the very last bit (that is not checked) can potentially flip the answer. Thus AND-function has a deterministic query complexity of n.

In communication complexity model there are two players Alice and Bob who get two n-bit inputs x and y respectively. Their goal is to compute a predefined function $f(x, y)$. As the players only know their part of the whole input (x, y), in order to compute the function $f(x, y)$ they will have to communicate[1]. We are interested in the minimum amount of communication needed to accomplish the task for all possible inputs. For example, consider a situation where Alice and Bob both want to fix a meeting with each other. They both have an n-bit binary string where a 0 entry at the i-th location of the string denotes that he/she is not free at that particular i-th hour. So their goal is to find a position j such that both have 1 at the j-th place of their inputs. The question is how many bits do they need to communicate before they can find a common slot for meeting. In the literature the decision version of this problem is called the set disjointness problem. It asks whether there is a common slot for meeting or not[2]. It can be proved that in worst case they need

[1]The output of the function f typically depends on both inputs x and y.

[2]Note that the complexity of the decision version and functional version of this problem are linearly related.

to send $n+1$ bits to find the answer. In Chapter 3 and Chapter 4 we review these models in detail.

1.2 Contributions

This thesis has two parts. In the first part, we study graph properties in the query complexity model. In particular, first we focus on the subgraph isomorphism and subgraph homomorphism problem. The subgraph isomorphism problem is a generalization of the well-known graph isomorphism problem, where one asks whether a given graph H is isomorphic to a subgraph of another graph G. Several central computational problems for graphs such as containing a Hamiltonian cycle, containing a clique, containing a perfect matching can be formulated as instances of the Subgraph Isomorphism Problem by fixing H appropriately. The subgraph isomorphism problem is a NP-Hard problem. Formally it is defined as follows: Let H be a (non-empty) graph on n vertices (possibly with isolated vertices) and let G be an unknown input graph (on n vertices) given by query access to its edges, i.e., queries of the form "Is $\{i,j\}$ an edge in G?". We say $H \leq G$ if G contains H as a (not necessarily induced) subgraph. Let $f_H : \{0,1\}^{\binom{n}{2}} \to \{0,1\}$ be defined as: $f_H(G) = 1$ iff $H \leq G$.

We give new lower bound for the quantum query complexity of this problem. Subgraph Isomorphism problem falls into a very important general class of properties called the monotone graph properties. A property $\mathcal{P}$ is said to be *monotone increasing* if for every input $x \leq y$ we have $\mathcal{P}(x) \leq \mathcal{P}(y)$, where $x \leq y$ denotes $x_i \leq y_i$ for all i. Monotone decreasing property is defined in a similar way. A graph property $\mathcal{P}$ is defined to be a property preserved under all possible isomorphisms of a graph G. In other words, a graph property $\mathcal{P}$ is a property of graphs that depends only on the abstract structure of the graphs, and not on the actual representation, such as particular labelling of the graphs. For example, containing a triangle (as a subgraph) is a property of a graph G.

Understanding the query complexity of monotone graph properties has a long history. In the deterministic setting the Aanderaa-Rosenberg-Karp Conjecture asserts that one must query all the $\binom{n}{2}$ edges in the worst-case. The randomized complexity of monotone graph properties is conjectured to be $\Omega(n^2)$. Yao [Yao87] obtained the first super-linear lower bound in the randomized setting using graph packing arguments. Subsequently his bound was improved by King [Kin88] and later by Hajnal [Haj91]. The current best known bound is $\Omega(n^{4/3}\sqrt{\log n})$ due to Chakrabarti and Khot [CK01]. Moreover, O'Donnell, Saks, Schramm, and Servedio [OSSS05] also obtained

an $\Omega(n^{4/3})$ bound via a more general approach for monotone transitive functions. Friedgut, Kahn, and Wigderson [FKW02a] obtained an $\Omega(n/p)$ bound where the p is the critical probability of the property. In the quantum setting, Buhrman, Cleve, de Wolf and Zalka [BCdWZ99] were the first to study quantum complexity of monotone graph properties. Santha and Yao [SY] obtained an $\Omega(n^{2/3})$ bound for general monotone properties. Their proof follows along the lines of Hajnal's proof.

Dietmar [Die92] studied the subgraph isomorphism problem and obtained an $\Omega(n^{3/2})$ bound for the randomized query complexity of Subgraph Isomorphism. This is currently the best known bound for the Subgraph Isomorphism Problem.

Until very recently, it was believed that for all total functions the quantum query complexity is at least the square root of the randomized one [ABDK15]. In this work we address the quantum query complexity of the Subgraph Isomorphism Problem and obtain the square root of the current best randomized bound.

In particular we show that for any H,

$$Q(f_H) = \Omega\left(\sqrt{\alpha_H \cdot n}\right),$$

where α_H denotes the size of a maximum independent set of H on n vertices. For example when H is a single edge, α_H is $n-1$ and thus $Q(f_H) = \Omega(n)$.

Using $Q(f_H) = \Omega\left(\sqrt{\alpha_H \cdot n}\right)$ we derive a lower bound of $Q(f_H) = \Omega(n^{3/4})$ for any H. We also get lower bounds in terms of the average degree of the vertices of H, the chromatic number of H, and the critical probability [FKW02a] of H: $Q(f_H) = \Omega\left(\frac{n}{\sqrt{d_{avg}(H)}}\right)$, $Q(f_H) = \Omega\left(\frac{n}{\sqrt{\chi_H}}\right)$, $Q(f_H) = \Omega\left(\sqrt{\frac{n}{p}}\right)$.

We then study the 3-uniform hypergraph version of this problem and obtain an $\Omega(n^{4/5})$ lower bound which improves the $\Omega(n^{3/4})$ bound obtained (for the 3-uniform hypergraph) via the minimum certificate size.

The main difference between the previous work and this one is that all the previous work, including that of Santha and Yao [SY], obtained the lower bounds based on an embedding of a *tribes* function [BBC+01] on a large number of variables in monotone graph properties. Recall that the tribes function with parameters k and ℓ, is a function $T(k,\ell)$ on $k \cdot \ell$ variables defined as: $\bigvee_{i\in[k]} \bigwedge_{j\in[\ell]} x_{ij}$. This method yields a lower bound of $\Omega(k \cdot \ell)$ for the randomized query complexity and $\Omega(\sqrt{k \cdot \ell})$ for the quantum. We deviate from this line by embedding a threshold

function T_n^t instead of a tribes. Recall that $T_n^t(z_1, \ldots, z_n)$ is a function on n variables that evaluates to 1 if and only if at least t of the z_i's are 1. Since the randomized complexity of T_n^t is $\Theta(n)$, this does not give us any advantage for obtaining super-linear randomized lower bounds. However, it *does* yield an advantage for the quantum lower bounds as the quantum query complexity of T_n^t is $\Theta(\sqrt{t(n-t)})$ [BCdWZ99, Pat92]. Note that as this technique works only in the quantum setting, the randomized versions of our bounds remain intriguingly open.

We also study a closely related Subgraph Homomorphism Problem. A homomorphism from a graph H into another graph G is a function $h : V(H) \to V(G)$ such that: if $(u, v) \in E(H)$ then $(h(u), h(v)) \in E(G)$. We define the homomorphism function $f_{[H]}$ as follows: $f_{[H]}(G) = 1$ if and only if H admits a homomorphism into the graph G. Note that unlike the isomorphism, the homomorphism need not be an injective function from $V(H)$ to $V(G)$. We show that for any H, $Q(f_{[H]}) = \Omega(n)$. Also, for any 3-uniform hypergraph H on n vertices we show $Q(f_{[H]}) = \Omega(n^{3/2})$. All our lower bounds of the Subgraph Isomorphism Problem or Subgraph Homomorphism Problem hold for the approximate degree $\widetilde{\deg}(f)$. Note that approximate degree is a lower bound on the quantum query complexity [BBC+01, Amb03]. Please refer to Chapter 5 for more details and the proofs.

Node Query Setting

In the Subgraph Isomorphism Problem (as discussed before) a graph H is known, and access to another graph G is given only via queries to the edges of G. The goal is to figure out whether G contains H as its subgraph. Note that it does not have to be induced. Now it is reasonable to ask what happens if we change the type of query and instead of asking whether an edge $e_i, i \in \binom{n}{2}$ is present in G we ask whether a *vertex* $v_j, j \in [n]$ is present in G. Thus any vertex can either be *active* or *inactive*. Naturally the set of all active vertices in G forms an induced subgraph. We can then ask whether the induced subgraph contains the graph H as a subgraph or we can ask whether it has some property $\mathcal{P}$. Formally it is defined as follows: A graph $G = (V, E)$ and a property $\mathcal{P}$ are *fixed*. We have access to $S \subseteq V$ via queries of the form "Does i belong to S?". We are interested in the minimum number of queries needed in the worst case in order to determine whether $G[S]$ – the subgraph of G induced on S – satisfies property $\mathcal{P}$, which we denote by $cost(\mathcal{P}, G)$. Similar to the edge query setting this new setting is combinatorially very rich and it opens a new platform for asking interesting questions.

The node-query setting is also a natural abstraction of scenarios where one is interested in the

properties of subgraph induced by active nodes in a network. For example, consider a physical network with several nodes and links between the nodes. The underlying network could be the network of routers which are physically connected by wires. Some of the routers may go on and off over time. At any given time, we want to know whether a message can be sent between two specified nodes via the active routers. This problem can be formulated in our setting as whether the subgraph induced by the active routers has a path between two specified nodes s and t or not. In Chapter 6 we discuss similar other examples in more detail.

We call G the base graph for a property $\mathcal{P}$. We say that a vertex i of G is *relevant* for the property $\mathcal{P}$ if there exists some $S \subseteq V$ containing i such that exactly one of $G[S]$ and $G[S - \{i\}]$ satisfies the property $\mathcal{P}$. In a sense, the vertex i is influential for S. We say that the graph G is *relevant* for a property $\mathcal{P}$ if all its vertices are *relevant* for $\mathcal{P}$. Note that we are interested in the relevance of graphs because if there exists a vertex i of a graph G that is not relevant for a property $\mathcal{P}$ then in order to check if G satisfies the property $\mathcal{P}$, we don't ever have to query the vertex i. The presence or absence of the vertex i doesn't influence the outcome.

The *minimum possible cost of a property* $\mathcal{P}$, denoted by $min\text{-}cost_n(\mathcal{P})$, is defined as follows: $min\text{-}cost_n(\mathcal{P}) = \min_G\{cost(\mathcal{P}, G) \mid G \text{ is relevant for } \mathcal{P} \ \& \ |V(G)| = n\}$. Notice that in this definition we take the minimum over all possible graphs that are relevant. We can define the notion of $\mathcal{G}\text{-}min\text{-}cost(\mathcal{P})$ where we restrict ourself to only a certain class of graphs $\mathcal{G}$. For example, we can consider the set of planar graphs or graphs with a certain amount of symmetry and ask what is the minimum cost of finding a property $\mathcal{P}$, $\mathcal{G}\text{-}min\text{-}cost(\mathcal{P})$.

Formally, $\mathcal{G}\text{-}min\text{-}cost_n(\mathcal{P}) = \min_{G \in \mathcal{G}}\{cost(\mathcal{P}, G) \mid G \text{ is relevant for } \mathcal{P} \ \& \ |V(G)| = n\}$.

Similar to the edge query model, there are several lines of investigation that can be done in the node query setting. We choose to focus on the effect of removing symmetry from graphs and study how low can the query complexity fall if we take away the symmetry assumption from the input graph. By taking away the symmetry we mean, first we ask what is the $\mathcal{G}\text{-}min_n\text{-}cost(\mathcal{P})$ when $\mathcal{G}$ is a set of graphs with a certain amount of symmetry. Then we compare it with the $\min\text{-}cost_n(\mathcal{P})$ where there is no restriction on the $\mathcal{G}$.

Recall that, a graph $G = (V, E)$ is called symmetric if given any two edges (u_1, v_1) and (u_2, v_2) of the graph G there is an automorphism A from the vertex set V to V such that $A(u_1) = u_2$ and $A(v_1) = v_2$. There are many different graph families defined by their automorphisms. The class of complete graphs has the highest amount of symmetry and the class of transitive graphs have the weakest form of symmetry [Sym]. Recall that a graph is transitive (or vertex-transitive) if its

automorphism group acts transitively upon its vertices.

Understanding the effect of symmetry on computation is a very well-studied theme in the literature. In the context of query complexity, there has been a substantial amount of effort invested in understanding the role of symmetry on computation. A recurrent theme has been to exploit the symmetry and/or some other structure [KS13] of the underlying functions to prove good lower bounds on their query complexity.

One such line of research concerns with the famous Andera-Rosenberge-Karp Conjecture [KSS84] which asserts that every non-trivial monotone graph property of n vertex graphs (in the edge-query model) must be evasive, i.e., its query complexity is $\binom{n}{2}$. While a weaker bound of $\Omega(n^2)$ is known, the conjecture remains widely open to this date. Even the randomized query complexity of monotone graph properties is also conjectured to be $\Omega(n^2)$ [FKW02b] and we are far from proving it. Thus it make sense to try to prove the conjectures for a certain class of graphs, for example graphs with symmetry. Sun,Yao, and Zhang [SYZ04a] studied query complexity of graph properties and several transitive functions in the edge query setting. Their motivation was to investigate how low can the query complexity fall if we drop the assumption of monotonicity or lower the amount of symmetry. In this work, we follow their footsteps and ask what happen if we *totally* drop the symmetry assumption. In the past, Lovász has also conjectured [Iva] that checking independence of S exactly in the node query setting is evasive.

Now it turns out that in the node query setting, when $\mathcal{G}$ has the highest amount of symmetry, i.e., when $\mathcal{G}$ is the class of complete graphs, then for every hereditary property $\mathcal{P}$, $\mathcal{G}\text{-}min\text{-}cost(\mathcal{P})$ is nearly the worst possible, i.e., $\Omega(n)$. Recall that a hereditary property is a property of graphs, which is closed under deletion of vertices as well as edges. For instance acyclicity, bipartiteness, planarity, and triangle-freeness are hereditary properties whereas connectedness and containing a perfect matching are not. Every hereditary property can be described by a (not necessarily finite) collection of its forbidden subgraphs. In our setting, every hereditary property is a monotone Boolean function. It also turns out that one does not require the whole symmetry of the complete graph to guarantee the $\Omega(n)$ bound. Even for weaker symmetry assumptions on graphs i.e., when $\mathcal{G}$ is the class of transitive graphs, for any hereditary property $\mathcal{P}$ the $\mathcal{G}\text{-}min\text{-}cost(\mathcal{P})$ would remain the highest possible, i.e., n. So for the complexity to fall down substantially we let go of the transitivity of $\mathcal{G}$ and take $\mathcal{G}$ to be the class of all graphs: $\mathcal{G}\text{-}min\text{-}cost(\mathcal{P}) = min\text{-}cost(\mathcal{P})$.

So then the question is how low can $min\text{-}cost(\mathcal{P})$ fall in the absence of any symmetry? This is the main question addressed by our work. We show that for any hereditary property $\mathcal{P}$, the

$min\text{-}cost(\mathcal{P})$ falls down at least quadratically, i.e., to $O(\sqrt{n})$. In particular we show that for any hereditary graph property $\mathcal{P}$: $min\text{-}cost(\mathcal{P}) = O(n^{1/(d_{\mathcal{P}}+1)})$, where $d_{\mathcal{P}}$ is the minimum degree of the minimum forbidden subgraph of the property $\mathcal{P}$.

The main question left open by our work is: does there exist a hereditary property $\mathcal{P}$ for which $min\text{-}cost(\mathcal{P})$ is exponentially low? In other words:

Question 1.2.1. *Is it true that for every hereditary property $\mathcal{P}$ there exists an integer $k_{\mathcal{P}} > 0$ such that*

$$min\text{-}cost(\mathcal{P}) = \Omega(n^{1/k_{\mathcal{P}}})?$$

As a partial answer to this question we prove that, when H is a graph on k vertices and $\mathcal{P}_H$ denotes the property of containing H as a subgraph, then, $min\text{-}cost(\mathcal{P}_H) = \Omega(n^{1/k})$. Furthermore, a non-constant lower bound holds for *any* hereditary property i.e., for any hereditary graph property $\mathcal{P}$, $min\text{-}cost(\mathcal{P}) = \Omega\left(\frac{\log n}{\log\log n}\right)$.

We study several other specific properties like Triangle-freeness, Independence etc. and proved tight bounds. We also consider restricted graph classes and prove several new results. A summary of these results can be found in Table 6.1. Please refer to Chapter 6 for the proofs and many more related results, which we have omitted in this section.

Communication Protocols

The second part of the thesis is about communication complexity and communication protocols. In this part we investigate the power of quantum proofs and quantum messages in the communication complexity settings. We also study a communication model called the Garden-hose model and explore its relation to other complexity models.

The Power of Quantum Messages

In Chapter 7 we investigate the relationship between quantum and classical one-way protocols. In this model two players Alice and Bob receive their corresponding inputs x and y respectively. Alice sends a message to Bob in order to compute the value of a function $f : \{0,1\}^n \times \{0,1\}^m \to \{0,1\}$. When we add quantum resources in, Alice communicates a quantum state and Bob performs a measurement, both depending on their respective inputs. They can have shared entanglement which

helps in their computation. Though very simple, this scenario is not at all fully understood. In fact the largest complexity gap between quantum and classical protocols of this kind for computing a total Boolean function is a factor of 2, as shown by Winter [Win04]. But on the other hand, all we know is that there could be examples where the gap is exponential. And indeed it is for certain partial functions (i.e., functions that are only defined on a subset of $\{0,1\}^n \times \{0,1\}^m$) [GKK+08].

However, an interesting bound on such speed-ups can be given when we investigate the effect of replacing quantum messages by classical messages. Suppose a total Boolean function f has a quantum one-way protocol with communication c, i.e., Alice sends c qubits to Bob, who can decide f with error $1/3$ by measuring Alice's message. We allow Alice and Bob to share an arbitrary input-independent entangled state. Extending Nayak's random access code bound [Nay99] Klauck [Kla00] showed that $Q^{A\to B,*}(f) \geq \Omega(VC(f))$, where $Q^{A\to B,*}(f)$ denotes the entanglement-assisted quantum one-way complexity of f, and $VC(f)$ the Vapnik-Chervonenkis dimension of the communication matrix of f. Together with Sauer's Lemma [Sau72] this implies that $D^{A\to B}(f) \leq O(Q^{A\to B,*}(f)\cdot m)$, where m is the length of Bob's input. A result such as this is much more interesting in the case of partial functions. Aaronson showed a weaker result for partial functions in the following two ways: for all partial Boolean functions $f : \{0,1\}^n \times \{0,1\}^m \to \{0,1\}$, $D^{A\to B}(f) \leq O(Q^{A\to B}(f) \cdot \log(Q^{A\to B}(f)) \cdot m)$ [Aar05] and $R^{A\to B}(f) \leq O(Q^{A\to B}(f) \cdot m)$ [Aar07].

By considering a more systematic progress measure than in Aaronson's proof, namely the relative entropy between the current guess and the target state and by using a different update rule for Bob's guess states we get the following improvement: $D^{A\to B}(f) \leq O(Q^{A\to B,*}(f) \cdot m)$ for all partial Boolean functions $f : \{0,1\}^n \times \{0,1\}^m \to \{0,1\}$. We also allow the quantum communication complexity on the right hand side to feature prior entanglement between the players Alice and Bob. Although an improvement by a mere logarithmic factor might seem unimportant, we note that having tight bounds for such basic questions is generally desirable, e.g., Nayak's bound for random access codes [Nay99] is an improvement by a logarithmic factor over previous work.

The Power of Quantum Proofs

We also study the power of quantum proofs. This is philosophically interesting because interactive proof systems are a fundamental concept in computer science. Quantum proofs have a number of disadvantages: reading them may destroy them, errors may occur during verification, verification

needs some sort of quantum machine, and it may be much harder to provide them than classical proofs. In the fully interactive setting Jain et al. [JJUW11] showed that the set of languages recognizable in polynomial time with the help of a quantum prover is equal to the set where the prover and verifier are classical (i.e., IP=QIP [JJUW11]). Thus in fully interactive setting quantum proofs do not give any advantages. So a natural question is what happens when we take away the interaction. Can quantum proofs be verified using fewer resources than classical proofs? Until now such a hope has not been verified formally. Aharonov and Naveh [AN02] first asked whether quantum proofs can ever be easier to verify than classical proofs (both times by quantum machines) in the absence of interaction, i.e., whether the class QMA is larger than its analogue with classical proofs but quantum verifiers, known as QCMA. An indication that quantum proofs may be powerful was given by Watrous [Wat00], who described an efficient QMA black-box algorithm for deciding nonmembership in a subgroup. Aaronson and Kuperberg [AK07], however, later showed how to solve the same problem efficiently using a classical witness, giving a QCMA black-box algorithm for the problem. They also introduced a quantum problem, for which they show that QMA black-box algorithms are more efficient than QCMA black-box algorithms. Using a quantum problem to show hardness for algorithms using classical proofs seems unfair though, and a similar separation has remained open for Boolean problems.

We compare the two modes of noninteractive proofs and quantum verification for a Boolean function in the setting of one-way communication complexity and show that quantum proofs are exponentially more powerful than the classical proofs in noninteractive setting. We do this by exhibiting a partial Boolean function f, such that the following holds: f can be computed in a protocol where a prover who knows x, y can provide a quantum proof to Alice, and Alice sends quantum message to Bob, such that the total message length (proof plus the message from Alice to Bob) is $O(\log n)$. And in the setting where a prover Merlin (knowing all inputs) sends a classical proof to Alice, who sends a quantum message to Bob, the total communication is $\Omega(\sqrt{n}/\log n)$. Thus there is a partial Boolean function f such that $QMA^{A\to B}(f) = O(\log n)$, while $QCMA^{A\to B,*}(f) = \Omega(\sqrt{n}/\log n)$. Note that this is the first known exponential gap between computing a Boolean function in a QCMA and a QMA-mode in any model of computation.

Garden-Hose Protocols

We also investigate how computation is affected if we restrict ourself to a certain class of communication protocols – in particular we focus on the garden-hose game, which is a memoryless and

reversible communication model. Chapter 8 contains all the detailed results.

Recently, Buhrman et al. [BFSS13] proposed a new measure of complexity for finite Boolean functions, called *the garden-hose complexity*. A garden-hose protocol works as follows: Two player Alice and Bob receive their respective inputs x and y. There are s shared pipes. Alice takes some pieces of hose and connects pairs of the open ends of the s pipes. She may keep some of the ends open. Bob acts in the same way for his end of the pipes. The connections Alice and Bob place depend on their local inputs x, y, and we stress that every end of a pipe is only connected to at most one other end of a pipe (meaning no Y-shaped pieces of hose may be used to split or combine flows of water). Finally, Alice connects a water tap to one of those open ends on her side and starts the water. Based on the connections of Alice and Bob, water flows back and forth through the pipes and finally ends up spilling on one side. If the water spills on Alice's side we define the output to be 0. Otherwise, the water spills on Bob's side and the output value is 1. It is easy to see that due to the way the connections are made the water must eventually spill on one of the two sides, since cycles are not possible. We say that a garden-hose protocol computes $f(x, y)$ if for all x, y water spills on Alice's side iff $f(x, y) = 0$. The garden-hose complexity of f, denoted by $GH(f)$, is the smallest s such that a garden-hose protocol of size s exists that computes f.

Garden-hose complexity can be viewed as a natural measure of space, in a situation where two players with private inputs compute a Boolean function cooperatively. Space-bounded communication complexity has been investigated before [BTY94, Kla04, KŠdW07], and recently Brody et al. [BCP+13] have studied a related model of space bounded communication complexity for Boolean functions (see also [PSS14]). In this context the garden-hose model can be viewed as a memoryless model of communication that is also reversible. A public coin randomized and a quantum version of the garden-hose model has also be defined [BFSS13].

We prove a number of results in this model. Buhrman et al. [BFSS13] proved that deterministic one-way communication complexity can be used to show lower bounds of up to $\Omega(n/\log n)$ for many functions. We improve on this by showing that non-deterministic communication complexity gives lower bounds on the garden-hose complexity of any function f. As an application, consider the function $IP(x, y) = \sum_{i=1}^{n}(x_i \cdot y_i)\ mod\ 2$. It is well known that $N(IP) \geq n + 1$ [KN97], hence we get that $GH(IP) \geq n$. The same bound holds for Disjointness.

Buhrman et al. [BFSS13] also showed that any one way communication complexity protocol with complexity $D_1(f)$ can be converted to a garden-hose protocol with $2^{D_1(f)} + 1$ pipes. One-way communication complexity can be much larger than two-way communication [PS84]. It turns

out that any 2-way deterministic communication protocol can also be converted to a garden-hose protocol so that the complexity $GH(f)$ is upper bounded by the *size* of the protocol tree of the communication protocol. Thus we show that for any function f, the garden-hose complexity $GH(f)$ is upper bounded by the number of edges in a protocol tree for f.

We then turn to compare the garden-hose model to another non-uniform notion of space complexity, namely branching programs. We show how to convert any *permutation branching program* to a garden-hose protocol with only a constant factor loss in size. In fact the inputs to the branching program we simulate can be functions (with small garden-hose complexity) instead of just variables. This allows us to use composition. In particular we show that $\mathrm{GH}(g(f_1, f_2, ..., f_k)) = O(s \cdot \max(C_i)) + O(1)$, where the permutation branching program size of g, $PBP(g) = s$ and $GH(f_i) = C_i$ and $f_i : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$. The f_i do not necessarily have the same inputs x, y.

The most important application of this simulation is that it allows us to find a garden-hose protocol for the distributed Majority function, $DMAJ(x, y) = 1$ *iff* $\sum_{i=1}^{n}(x_i \cdot y_i) \geq \frac{n}{2}$ where $x, y \in \{0,1\}^n$, that has size $O(n \cdot \log^3 n)$, disproving the conjecture in [BFSS13] that this function has complexity $\Omega(n^2)$.

We also relate $GH(f)$ to the formula size of f. To do so we examine composition of garden-hose protocols by popular gate functions. In particular we show that for $(f_1, f_2, .., f_k)$, where each function f_i has garden-hose complexity $GH(f_i)$: i) $GH(\bigvee f_i) = O(\sum GH(f_i))$, ii) $GH(\bigwedge f_i) = O(\sum GH(f_i))$, iii) $GH(\oplus f_i) = O(\sum GH(f_i))$, iv) $GH(MAJ(f_i)) = O(\sum GH(f_i) \cdot \log^3 k)$.

Then by using the fact that formulae over the set of all fan-in 2 function can be efficiently simulated by branching programs (by Giel [Gie01]) we show the following:

Let F be a formula for a Boolean function g on k inputs made of gates $\{\wedge, \vee, \oplus\}$ of arbitrary fan-in. If F has size s and $GH(f_i) \leq c$ for all i, then for all constants $\epsilon > 0$ we have $GH(g(f_1, f_2, .., f_k)) \leq O(s^{1+\epsilon} \cdot c)$.

In particular it implies that when the f_i's are single variables $GH(g) \leq O(s^{1+\epsilon})$ for all constants $\epsilon > 0$. Thus any lower bound on the garden-hose complexity of a function g yields a slightly smaller lower bound on formula-size (all gates of fan-in 2 allowed). The best lower bound of $\Omega(n^2/\log n)$ known for the size of formulae over the basis of all fan-in 2 gate function is due to Nečiporuk [Nec66]. The Nečiporuk lower bound method (based on counting subfunctions) can also be used to give the best general branching program lower bound of $\Omega(n^2/\log^2 n)$ (see [Weg87]). Thus our result implies any lower bound larger than $\Omega(n^{2+\epsilon})$ for the garden-hose

model would immediately give lower bounds of almost the same magnitude for formula size and permutation branching program size. But proving super-quadratic lower bounds in these models is a long-standing open problem. Buhrman et al. [BFSS13] argued that a super-polynomial lower bound for the garden-hose complexity of any explicit function f would be difficult to prove. Our result yields that even getting larger than $\Omega(n^{2+\epsilon})$ would be difficult for any explicit function f.

In other results we also define private coin randomized garden-hose protocols and time bounded garden-hose model. Buhrman et al. [BFSS13] have shown how to de-randomize a public coin protocols at the cost of increasing size by a factor of $O(n)$, so the factor n in the separation between public coin and deterministic protocols above is the best that can be achieved. This raises the question whether private coin protocols can ever be more efficient in size than the optimal deterministic protocol. We show that there are no very efficient private coin protocols for the Equality function: $RGH^{pri}(Equality) = \Omega(\sqrt{n}/\log n)$.

In the time-bounded garden-hose complexity of a function f, where we restrict the number of times water can flow through pipes to some value k, we have $GH_k(f) = \Omega(2^{D_k(f)/k})$, where GH_k denotes the time-bounded garden-hose complexity, and D_k the k-round deterministic communication complexity. This result leads to strong lower bounds for the time bounded complexity of e.g. Equality, and to a time-hierarchy based on the pointer jumping problem.

References

This thesis is based on the following papers.

- *Klauck, H., and Podder, S. (2014). Two results about quantum messages. In Proceedings of the 39th International Symposium on the Mathematical Foundations of Computer Science (MFCS) 2014. Springer Berlin Heidelberg, 2014. Pages 445-456. [KP14b]*

- *Klauck, H., and Podder, S. (2014). New Bounds for the Garden-Hose Model. Foundation of Software Technology and Theoretical Computer Science. 2014, Page 481. LIPIcs-Leibniz International Proceedings in Informatics. Vol. 29. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2014. [KP14a]*

- *Kulkarni, R., and Podder, S. (2016). Quantum Query Complexity of Subgraph Isomorphism and Homomorphism. In Proceedings of the 33rd Symposium on Theoretical Aspects of Computer Science (STACS 2016), Pages 48:1–48:13 [KP16]*

- *Balaji, N., Datta, S., Kulkarni, R., and Podder, S. (2016). Graph properties in node-query setting: effect of breaking symmetry. In Proceedings of the 41st International Symposium on the Mathematical Foundations of Computer Science (MFCS) 2016. [BDKP15]*

Chapter 2

Basics of Quantum Computation and Information

2.1 Introduction

In this chapter we review some relevant aspects of quantum computing and quantum information which are essential for this thesis. In Section 2.2 the concepts of Hilbert space, quantum states and different quantum operators are introduced. Section 2.2.6 reviews how to represent quantum states as density operators. Finally in Section 2.3 we explore some quantum information measures and properties which will be useful later on.

2.2 Quantum Mechanics

All form of digital electronic computers that we use today are governed by classical physics. But when we consider tiny atomic level systems, Newtonian mechanics, which explains the large scale systems quite well, fails to explain phenomena properly. A new form of physics called "quantum physics", which began to be investigated around 1900, turns out to give much better explanations of small scale systems. Quantum physics shows that the world actually behaves quite counter-intuitively. For instance, according to classical mechanics any system can only be in any one of the many possible states at any given time. Whereas quantum physics tells us that this intuition is wrong and a system can be simultaneously in a superposition of many possible states at the same time.

Quantum computation is the branch of computer science which tries to use properties of

quantum mechanics to develop new quantum algorithms that would outperform any classical algorithm to solve certain tasks. Similarly, quantum information theory is a branch of information theory that generalizes classical information theory to the quantum world. It aims to understand how information is stored or transmitted efficiently in a quantum system. Here in this chapter we give a brief introduction to all the necessary topics. Reader may refer to the lecture notes by Watrous [Wat06, Wat11] and the book by Nielsen and Chuang [NC10].

When we consider a single classical bit, it can be either in 0 or 1 state. A single quantum bit or qubit, however, can be in a superposition of both 0 and 1 state at the same time,

$$|\phi\rangle = \alpha_0|0\rangle + \alpha_1|1\rangle.$$

Here $\forall i, \alpha_i$ is a complex number and $\sum_i |\alpha_i|^2 = 1$. The notation $|.\rangle$, used above, is called the Direc "ket" notation, named after its inventor Paul Dirac.

Similarly, a two bit quantum state $|\phi\rangle$ can be in a superposition of four different classical states:

$$|\phi\rangle = \alpha_{00}|00\rangle + \alpha_{01}|01\rangle + \alpha_{10}|10\rangle + \alpha_{11}|11\rangle,$$

such that $|\alpha_{00}|^2 + |\alpha_{01}|^2 + |\alpha_{10}|^2 + |\alpha_{11}|^2 = 1$.

In general a physical system can consist of N different classical states $|1\rangle, \cdots, |N\rangle$. Then a quantum state $|\phi\rangle$ is a superposition of all such N different possible states:

$$|\phi\rangle = \alpha_1|1\rangle + \cdots + \alpha_N|N\rangle, \text{ where } \sum_{i=1}^{N} |\alpha_i|^2 = 1.$$

2.2.1 Qubits as Vectors in Complex Euclidean Spaces

Let's denote by $\mathbb{C}^S$ the set of all possible functions from a set S of size N to $\mathbb{C}$. Then each such function can be thought of as an N dimensional complex vector and $\mathbb{C}^S$ forms a vector space of dimension N. For example when $S = \{0,1\}^n$ we have a 2^n dimensional complex vector space $\mathbb{C}^{\{0,1\}^n}$ where the vectors $\{|x\rangle \mid x \in \{0,1\}^n\}$ form an orthogonal basis of the vector space.

A quantum state $|\phi\rangle$ can be thought of as a *unit* vector of dimension N sitting in an N dimensional complex Euclidean vector space (equipped with an inner product). A finite dimensional vector space equipped with an inner product is called a finite dimensional Hilbert space. And we only use complex/real euclidean spaces in the thesis.

Thus any quantum state $|\phi\rangle = \sum_i^N \alpha_i |i\rangle$ can be viewed as a unit vector:

$$|\phi\rangle = \begin{pmatrix} \alpha_1 \\ \alpha_2 \\ \vdots \\ \alpha_N \end{pmatrix}. \tag{2.1}$$

For example, the smallest non-trivial Hilbert space has dimension 2. There can be several ways of choosing an orthogonal basis for this vector space. One such basis is the set of unit vectors $\{|0\rangle, |1\rangle\}$. Then a single bit quantum state is a unit vector $\alpha_0|0\rangle + \alpha_1|1\rangle$ in this Hilbert space, satisfying $|\alpha_0|^2 + |\alpha_1|^2 = 1$.

In this thesis we represent quantum states with vectors. The dimension of a vector representing a qubit system grows exponentially with the number of qubits. The Dirac notation is often very useful to represent high dimensional quantum systems.

$$|0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \tag{2.2}$$

$$|1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix} \tag{2.3}$$

For every ket $|\phi\rangle$ there is a bra $\langle\phi|$. The bra is defined as the conjugate transpose (or adjoint) (see Section 2.2.5 for the definition) of $|\phi\rangle$, i.e., $\langle\phi| = (|\phi\rangle)^\dagger$.

For example, if the quantum state $|\phi\rangle$ is the following:

$$|\phi\rangle = \begin{pmatrix} \frac{1}{\sqrt{2}} \\ \frac{1-i}{\sqrt{2}} \end{pmatrix} \tag{2.4}$$

then its bra $\langle\phi|$ is,

$$\langle\phi| = \begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{1+i}{\sqrt{2}} \end{pmatrix}. \tag{2.5}$$

2.2.2 Tensor Product, Entangled States

Tensor product of vector spaces

Let's say we have two vector spaces $\mathbb{C}^S$ and $\mathbb{C}^{S'}$ of dimension n and m respectively. Then the tensor product of these two vector spaces denoted by $\mathbb{C}^S \otimes \mathbb{C}^{S'}$ is an nm dimensional vector space.

Tensor product is useful when we want to combine two separate quantum systems and consider them together. For example consider a single qubit $|\phi\rangle = \alpha_0|0\rangle + \alpha_1|1\rangle$ which is a unit vector in a 2-dimensional Hilbert space U. Consider another single qubit $|\psi\rangle = \beta_0|0\rangle + \beta_1|1\rangle$ which is a unit

vector in another 2-dimensional Hilbert space V.

Now the state $|\phi\rangle \otimes |\psi\rangle = |\phi\rangle|\psi\rangle = |\phi\psi\rangle = \alpha_0\beta_0|0\rangle|0\rangle + \alpha_0\beta_1|0\rangle|1\rangle + \alpha_1\beta_0|1\rangle|0\rangle + \alpha_1\beta_1|1\rangle|1\rangle$ (where $\sum_{i,j} |\alpha_i\beta_j|^2 = 1$) is a combined state that is in Hilbert space $U \otimes V$.

However not all quantum states can be written as a tensor product. For instance consider the following states:

$$|\phi^{\pm}\rangle = \frac{1}{\sqrt{2}}|00\rangle \pm \frac{1}{\sqrt{2}}|11\rangle.$$

These are examples of entangled states. This type of states are unique to quantum mechanics and allow us to perform operations such as teleportation which have no classical analogue.

2.2.3 Inner product, Euclidean Norm, ℓ-Norm

Inner Product: Given two vectors $|\phi\rangle, |\psi\rangle \in \mathbb{C}^S$ where $|S| = N$, the inner product of these two vectors are defined as $\langle\phi|\psi\rangle = \sum_{i\in S} \bar{\phi_i}\psi_i$. Here by ϕ_i we mean the i-th component of the vector $|\phi\rangle$. And $\bar{\phi_i}$ represents the conjugate transpose of the i-th component, ϕ_i.

Euclidean Norm: The Euclidean norm of $|\phi\rangle$ is defined by

$$|||\phi\rangle|| = \sqrt{\langle\phi|\phi\rangle} = \sqrt{\sum_{i\in S} |\phi_i|^2}.$$

ℓ_p-Norm: The ℓ_p-Norm of a vector $|\phi\rangle$ is defined by

$$|||\phi\rangle||_p = \left(\sqrt{\sum_{i\in S} |\phi_i|^p}\right)^{1/p}.$$

Thus ℓ_2 norm is the Euclidean Norm. And $|||\phi\rangle||_\infty = max\ \{\phi_i | i \in S\}$.

2.2.4 Orthogonality and Orthonormality

Orthogonality: A finite non-empty set S of vectors $\{|\phi^i\rangle | i \in S\}$ form an orthogonal set if the inner product of any two vectors is 0, i.e., $\langle\phi^i|\phi^j\rangle = 0$ for all $i \neq j \in S$.

Basis Vectors: A set of vectors $\{|\phi^i\rangle | i \in S\}$ in a vector space V is said to form a basis, or is called a set of basis vectors, if the vectors $\{|\phi^i\rangle | i \in S\}$ are linearly independent and every other vector in the vector space is a linear combination of the vectors from the set $\{|\phi^i\rangle | i \in S\}$.

Orthonormality: An orthogonal set of vectors $\{|\phi^i\rangle | i \in S\}$ is called orthonormal set if all vectors $|\phi^i\rangle, i \in S$ are unit vectors.

When an orthonormal set forms a basis for a vector space V it is called a orthonormal basis. An orthonormal set $\{|\phi^i\rangle | i \in S\}$, in some $\mathbb{C}^{S'}$, forms a basis for $\mathbb{C}^{S'}$ iff the cardinalities of S and S' are same, i.e., $|S| = |S'|$.

2.2.5 Linear Operators

Linear Operator and Linear Matrix: A linear operator $L : \mathbb{C}^S \to \mathbb{C}^{S'}$ between two vector spaces $\mathbb{C}^S$ and $\mathbb{C}^{S'}$ is a linear mapping from $\mathbb{C}^S$ to $\mathbb{C}^{S'}$. Note that a function $f : A \to B$ is called a linear map if for any two vectors $x, y \in A$ and any complex number α, $f(x + y) = f(x) + f(y)$ and $\alpha f(x) = f(\alpha x)$.

Let $|\phi\rangle$ be described in terms of the basis vectors $\{|\phi^i\rangle\}$. $|\phi\rangle = \sum_i \alpha_i |\phi^i\rangle$. Then the operator L is:

$$L\left(\sum_i \alpha_i |\phi^i\rangle\right) = \sum_i \alpha_i L(|\phi^i\rangle).$$

When a linear operator is from $\mathbb{C}^S$ to $\mathbb{C}^S$, we say it is defined on $\mathbb{C}^S$. We denote by $L(\mathbb{C}^S)$ the set of all linear operators from $\mathbb{C}^S$ to $\mathbb{C}^S$. For any vector space $\mathbb{C}^S$ an identity operator maps all elements $|\phi\rangle \in \mathbb{C}^S$ to $|\phi\rangle$ itself. The zero operator maps every element of $\mathbb{C}^S$ to the zero vector.

For a fixed basis any linear operator can be represented by a matrix. For instance, a linear operator $L : \mathbb{C}^S \to \mathbb{C}^{S'}$ where $|S| = n$ and $|S'| = m$ can be represented by an $m \times n$ complex matrix $L'_{m \times n}$. Note that if we multiply this matrix with a vector $|\phi\rangle$ from $\mathbb{C}^S$, this matrix performs the linear operation L on the vector $|\phi\rangle$ and transform it to an element, $L|\phi\rangle$, of the vector space $\mathbb{C}^{S'}$.

Now we go into the details of some different types of operators that are useful for quantum computation and information. But before that we briefly review conjugate and transpose of a matrix.

Conjugate, Transpose, and Adjoint

Transpose of a Matrix, M^T: Transpose of a Matrix M is a matrix M^T which is obtained by changing the columns of M with rows and rows of M with the column, i.e., any ij-th entry of the matrix, A_{ij} becomes A_{ji} for all i and j. For example,

$$M = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} \textit{ becomes } M^T = \begin{bmatrix} 1 & 4 & 7 \\ 2 & 5 & 8 \\ 3 & 6 & 9 \end{bmatrix}$$

Conjugate of a Matrix, $\bar{M}$: Conjugate of a Matrix M is a matrix $\bar{M}$ which is obtained by changing each entry of the matrix M to its complex conjugates.

For example,

$$M = \begin{bmatrix} 1 & \frac{1}{\sqrt{2}} & 3 \\ 4 & 5 & 6 \\ 7 & 8 & \frac{1-i}{\sqrt{2}} \end{bmatrix} \textit{ becomes } M^T = \begin{bmatrix} 1 & \frac{1}{\sqrt{2}} & 3 \\ 4 & 5 & 6 \\ 7 & 8 & \frac{1+i}{\sqrt{2}} \end{bmatrix}$$

Adjoint of a Matrix, $M^\dagger$: Adjoint of a Matrix M is a matrix $M^\dagger$ which is obtained by taking both the transpose and conjugate of the matrix M.

Normal Operators: An operator $M : \mathbb{C}^S \to \mathbb{C}^S$ is normal if and only if it commutes with its adjoint $M^\dagger$, i.e., $MM^\dagger = M^\dagger M$.

Unitary Operators: A normal operator $M : \mathbb{C}^S \to \mathbb{C}^S$ is called a unitary operator if and only if $MM^\dagger = M^\dagger M = I$.

Hermitian operators: Hermitian operators are a subsets of the Normal operators. An operator $M : \mathbb{C}^S \to \mathbb{C}^S$ is called hermitian if it is equivalent to its adjoint, i.e., $M = M^\dagger$. Hermitian operators have the property that their eigenvalues are all real numbers.

Positive Definite Operators, Positive Semi-definite Operators: An operator $M : \mathbb{C}^S \to \mathbb{C}^S$ is said to be a positive semi-definite operator, or PSD operator in short, if for all choice of non-negative vectors $|\phi\rangle \in \mathbb{C}^S$, $\langle\phi|M|\phi\rangle$ is a non-negative real number. Alternatively, an operator $M : \mathbb{C}^S \to \mathbb{C}^S$ is called a positive semi-definite operators if and only if M is Hermitian and all eigenvalues of A are non-negative real numbers.

An operator $M : \mathbb{C}^S \to \mathbb{C}^S$ is said to be a positive definite operator if in addition of being positive semi-definite it is also invertible. Alternatively, an operator $M : \mathbb{C}^S \to \mathbb{C}^S$ is called a positive definite operator if and only if M is Hermitian and all eigenvalues of A are positive [HJ85].

Density Operator

A positive semi-definite operator of trace 1 is called density operator.

The trace of a matrix $M_{n\times n}$ is defined to be the sum of all its diagonal elements.

$$Tr(M) = \sum_i M(i,i).$$

We refer the reader to an excellent lecture notes by Watrous [Wat11] for properties of different

operators.

2.2.6 Quantum State as a Density Matrix or Density Operator

To describe a quantum system using a density matrix, a quantum state in its vector representation $|\phi\rangle \in \mathbb{C}^S$ (where $|S| = N$) is represented by an $N \times N$ matrix $M = |\phi\rangle\langle\phi|$ (outer product).

For example, if we have a single qubit $|\phi\rangle = \alpha_0|0\rangle + \alpha_1|1\rangle$ then in density matrix notation it is represented as,

$$|\phi\rangle\langle\phi| = \begin{pmatrix} \alpha_0 \\ \alpha_1 \end{pmatrix} (\bar{\alpha_0} \quad \bar{\alpha_1}) = \begin{pmatrix} \alpha_0\bar{\alpha_0} & \alpha_0\bar{\alpha_1} \\ \alpha_1\bar{\alpha_0} & \alpha_1\bar{\alpha_1} \end{pmatrix}. \tag{2.6}$$

Given any quantum state in the vector notation this is how we obtain the density matrix or density operator form. Thus for any quantum state $|\phi\rangle$, its corresponding density matrix is of the form $|\phi\rangle\langle\phi|$. When a density matrix has $|\phi\rangle\langle\phi|$ form we call it a *pure state*. But note that not all density matrix have this form. Density matrices are capable of describing an ensemble or mixture of several quantum states. Below we describe this in detail.

Mixed State (Non-pure State)

A mixture or ensemble of several quantum states can be written as $\{p_i, |\phi_i\rangle\}$ where each quantum state $|\phi_i\rangle$ occurs with probability p_i. Naturally $\sum_i p_i = 1$.

The density matrix representing this mixture is

$$\rho = \sum_i p_i |\phi_i\rangle\langle\phi_i|.$$

Let us see the following two examples:

A mixed state ρ is defined as follows: With $1/2$ probability it is $|0\rangle$ and with $1/2$ probability it is $|1\rangle$.

Thus

$$\rho = 1/2|0\rangle\langle 0| + 1/2|1\rangle\langle 1|$$

$$= \frac{1}{2}\begin{pmatrix} 1 \\ 0 \end{pmatrix}(1 \quad 0) + \frac{1}{2}\begin{pmatrix} 0 \\ 1 \end{pmatrix}(0 \quad 1)$$

$$= \frac{1}{2}\begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix} + \frac{1}{2}\begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix} = \frac{1}{2}I.$$

Here I is a 2×2 identity matrix.

$|\pm\rangle = \frac{1}{\sqrt{2}}(0 \pm 1)$. Now consider another example where $\rho = 1/2|+\rangle\langle+| + 1/2|-\rangle\langle-|$.

Thus

$$\rho = 1/2|+\rangle\langle+| + 1/2|-\rangle\langle-|$$

$$= \frac{1}{2}\begin{pmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{pmatrix}\begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{pmatrix} + \frac{1}{2}\begin{pmatrix} \frac{1}{\sqrt{2}} \\ -\frac{1}{\sqrt{2}} \end{pmatrix}\begin{pmatrix} \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{pmatrix}$$

$$= \frac{1}{2}\begin{pmatrix} \frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} & \frac{1}{2} \end{pmatrix} + \frac{1}{2}\begin{pmatrix} \frac{1}{2} & -\frac{1}{2} \\ -\frac{1}{2} & \frac{1}{2} \end{pmatrix} = \frac{1}{2}I.$$

Density Matrix for Different States

Notice that in the above two examples, even thought, the ρs are two different probability distributions over two different states, their density matrices are same. Thus one density matrix can represent many different possible state mixtures. So the density matrix notation is a more unambiguous and more efficient description of a qubit system. If two different quantum ensembles have a same density matrix then there is no observable difference between these two quantum ensembles.

Note that the density matrix $\rho = \sum_i p_i|\phi_i\rangle\langle\phi_i|$ is a positive semidefinite matrix as for any non-negative $|\psi\rangle$,

$$\langle\psi|\rho|\psi\rangle = \langle\psi|\left(\sum_i p_i|\phi_i\rangle\langle\phi_i|\right)|\psi\rangle = \sum_i p_i\langle\psi|\phi_i\rangle\langle\phi_i|\psi\rangle = \sum_i p_i|\langle\psi|\phi_i\rangle|^2 \geq 0.$$

It is also easy to see that convex combinations preserve the Trace.

$$Tr(\rho) = Tr\left(\sum_i p_i|\phi_i\rangle\langle\phi_i|\right) = \sum_i p_i Tr(|\phi_i\rangle\langle\phi_i|) = \sum_i p_i Tr(\langle\phi_i||\phi_i\rangle)$$

$$= \sum_i p_i\langle\phi_i|\phi_i\rangle = 1.$$

2.2.7 Partial Trace

If we have a system of joint state space of $A = \mathbb{C}^S$ and $B = \mathbb{C}^{S'}$ and a density matrix ρ_{AB} describes the system. Then if we want to discard B and only consider A, the operation is called tracing out the space B.

$$\rho_A = Tr_B(\rho_{AB}) = \sum_{x \in S'} (I_{\mathbb{C}^S} \otimes \langle x|)\rho(I_{\mathbb{C}^S} \otimes |x\rangle).$$

In other word, a partial trace is a linear map from $\mathbb{C}^{S \cup S'} \to \mathbb{C}^S$ that is determined by the above equation. Note that the partial trace operation is not about destroying some part of the system. It is about consider a part of the system and solely focusing on that.

If we perform a partial trace on a product state $\rho_{AB} = \rho_A \otimes \rho_B$, then $Tr_B(\rho_{AB}) = \rho_A$. No information about the state in A is lost. However in the case of non-product states, information may be lost. For example, consider the density operator of the following Bell state:

$$|\phi^+\rangle = \frac{1}{\sqrt{2}}|0_A 0_B\rangle + \frac{1}{\sqrt{2}}|1_A 1_B\rangle.$$

$$\rho_{AB} = \frac{1}{2}\left(|0_A 0_B\rangle + |1_A 1_B\rangle\right) \otimes \left(\langle 0_A 0_B| + \langle 1_A 1_B|\right).$$

$$= \frac{1}{2}\left(|0_A 0_B\rangle\langle 0_A 0_B| + |0_A 0_B\rangle\langle 1_A 1_B| + |1_A 1_B\rangle\langle 0_A 0_B| + |1_A 1_B\rangle\langle 1_A 1_B|\right).$$

Now, if we take trace over B we will have,

$$\rho_A = Tr(\rho_{AB}) = \frac{1}{2}(|0_A\rangle\langle 0_A| + |1_A\rangle\langle 1_A|).$$

It is a maximally mixed state and if we perform the same operation for other Bell states i.e., $|\phi^-\rangle, |\psi^+\rangle$ or $|\psi^-\rangle$ we will obtain the same result. Thus the information about which state we have started with is lost.

2.2.8 Measurements

Quantum measurements are described by a collection of measurement operators $\{M_i \mid i \in S'\} \subset L(\mathbb{C}^S)$ which acts on the state space being measured. $\{M_i\}$ must satisfy the condition that the set S' can be any finite non-empty set and $\sum_{i \in S'} M_i^\dagger M_i = I$.

When a quantum system is in state ρ and being measured the measurement result will be one of the element of the set S' with probability $Tr(M_i \rho M_i^\dagger)$.

And after the measurements the remaining state (or the collapsed state) is $\frac{M_i \rho M_i^\dagger}{Tr(M_i \rho M_i^\dagger)}$.

2.3 Quantum Information

Now we briefly review some relevant tools and techniques of quantum information, which are needed later on. The theory of quantum information is a generalization of the classical information theory to the quantum world. The main goal of quantum information is to understand how is information stored in a quantum system and how efficiently can we transmit information using a quantum system.

2.3.1 Distance between Two Quantum States

In quantum information very often we want to know how far two given quantum states are?

Total Variation Distance

Classically, the total variation distance is a metric of how distinguishable one distribution is from another. Given two probability distributions $\{p_i\}$ and $\{q_i\}$ the *total variation distance* is defined as:

$$D(p_i, q_i) = \frac{1}{2}\sum_i |p_i - q_i|.$$

Similarly in quantum information, the trace distance is a metric of how distinguishable one density matrix is from another.

Before we define trace distance let us define the trace norm for any Hermitian operator.

Definition 2.3.1. *The trace norm of a Hermitian operator ρ is defined as $||\rho||_t = \mathsf{Tr}\sqrt{\rho\rho^\dagger}$.*

Definition 2.3.2. *Given two density matrices ρ and σ the trace distance between them is defined as follows:*

$$||\rho - \sigma||_t = \frac{1}{2}\mathsf{Tr}(\sqrt{(\rho-\sigma)^\dagger(\rho-\sigma)}).$$

Note that if ρ and σ commute then they are diagonal in the same basis, i.e., if $\rho = \sum_i \alpha_i|i\rangle\langle i|$ and $\sigma = \sum_i \alpha_i'|i\rangle\langle i|$ for some orthogonal basis $\{|i\rangle\}$, then their trace distance $||\rho - \sigma||_t = \frac{1}{2}\mathsf{Tr}(\sum_i(\alpha_i - \alpha_i')|i\rangle\langle i|)$ which is exactly the classical total variation distance between the eigenvalues of ρ and the eigenvalues of σ.

Fidelity

Fidelity is a measure of the "closeness" of two probability distributions.

Given two probability distributions $\{p_i\}$ and $\{q_i\}$ the fidelity between them is defined as:

$$F(p_i, q_i) = \sum_i \sqrt{p_i, q_i}.$$

It is very different from the trace distance. Unlike trace distance it is not a metric. Rather it has a nice geometric interpretation. Given two probability distribution $\{p_i\}$ and $\{q_i\}$, we can construct two vectors of unit size with component of $\sqrt{p_i}$ and $\sqrt{q_i}$. Then the fidelity is the inner product between these two vectors.

Similarly in quantum information theory, the fidelity between two density matrices ρ and σ is defined by,

$$F(\rho, \sigma) = \mathsf{Tr}(\sqrt{(\sqrt{\rho})\sigma(\sqrt{\rho})}).$$

Fidelity and the trace distance are related $(1 - F(\rho, \sigma)) \leq ||\rho - \sigma||_t \leq \sqrt{1 - F(\rho, \sigma)^2}$.

For pure states, $||\rho - \sigma||_t = \sqrt{1 - F(\rho, \sigma)^2}$.

2.3.2 Entropy

Entropy is one of the fundamental and most useful concepts in information theory. Before going into the quantum entropy, let's review the classical entropy. The reader may refer to a nice book by Cover and Thomas [CT12] for literature.

Shannon Entropy:

Shannon Entropy, denoted by $H(X)$, of a random variable X measures the amount of information conveyed by the value of the random variable X.

$$H(X) = -\sum_x p_x \log_2 p_x.$$

It is also the minimum number of physical resources required to store each values of the random variable X on average.

Relative Entropy:

Similar to Fidelity, the relative entropy captures how close two probability distributions $\{p_i\}$ and $\{q_i\}$ are.

$$H(p_i||q_i) = \sum_i p_i \log \frac{p_i}{q_i} = -H(X) - \sum_i p_i \log q_i.$$

The relative entropy is non-negative and $H(p_i||q_i) = 0$ iff $p_i = q_i, \forall i$. Note that relative entropy is not a metric and it is also not symmetric i.e., $H(p_i||q_i)$ may not be equal to $H(q_i||p_i)$.

Von Neumann Entropy:

The Shannon entropy deals with the uncertainty associated to a classical random variable. Von Neumann entropy captures entropy of quantum states.

Definition 2.3.3. *The Von Neumann entropy of a quantum state ρ is defined as,*

$$S(\rho) = -\mathsf{Tr}(\rho \log \rho).$$

The logarithm is over base 2. For the ease of calculation the above formula is re-written as follows [NC10]:

$$S(\rho) = -\sum_i \pi_i \log \pi_i.$$

Here π_i are the eigenvalues of the state ρ.

Relative Von Neumann Entropy:

The relative Von Neumann entropy of quantum states ρ, σ is defined as,

$S(\rho||\sigma) = \mathsf{Tr}\rho \log \rho - \mathsf{Tr}\rho \log \sigma$ *if supp* $\rho \subseteq$ *supp* σ,

otherwise, $S(\rho||\sigma) = \infty$.

Relative Min-Entropy:

The relative min-entropy of ρ, σ is defined as

$S_\infty(\rho||\sigma) = \inf\{c : \sigma - \rho/2^c$ *is positive semidefinite*$\}$.

It is easy to see that $S(\rho||\sigma) \leq S_\infty(\rho||\sigma)$, see [Dat09] for a proof.

Chapter 3

Query Complexity

3.1 Introduction

This chapter is about query complexity. In the next section we introduce query complexity and review different models of it. In Section 3.3 we explore various query complexity measures which are used to give lower bounds to different models of query complexity. In Section 3.4 we summarize the relation between different complexity measures and query complexity models.

3.2 Query Complexity Models

The decision tree model (aka the query model), perhaps due to its simplicity and fundamental nature, has been extensively studied in the past and still remains a rich source of many fascinating investigations. In this thesis we focus on Boolean functions, i.e., the functions of the form $f : \{0,1\}^n \to \{0,1\}$. This section reviews different models of query complexity.

3.2.1 Deterministic Query Model

A deterministic decision tree T_f for f takes $x = (x_1, \ldots, x_n)$ as an input and determines the value of $f(x_1, \ldots, x_n)$ using queries of the form "is $x_i = 1$?".

In any deterministic decision tree T_f for f each internal node is marked with some literal x_i, $i \in [n]$ and has two out-going edges labelled with 0 and 1. All the leaves have label 0 or 1. For any given input x the tree is evaluated from the root and depending on the value of the x_i of the root, either the left sub-tree or the right sub-tree of the root gets evaluated. This process continues until it reaches some leaf node. The output is the value of the leaf node reached. We say

a deterministic decision tree T_f computes a function f correctly if its output on x is same as $f(x)$ for all $x \in \{0,1\}^n$.

Formally a query can be thought of as a transformation of a three registers tuple (i,b,z), where i is a $\log n$-bit register that contains a pointer which can point to any location of the input x, b contains 0 and z is an m-bit register, called the workspace. The query operation O reads the value of i and adds the value x_i to the second register b. The query does not affect the workspace z.

$$O : (i, b, z) \rightarrow (i, b \oplus x_i, z).$$

The register z is used for internal computations which occur in between two queries.

Let $C(T_f, x)$ denote the cost of the computation, that is the number of queries made by T_f on an input x. The *deterministic decision tree complexity* (aka the deterministic query complexity) of f is defined as

$$D(f) = \min_{T_f} \max_{x} C(T_f, x).$$

Below are the examples of two decision trees one for computing the Equality function and another for the AND function (see Figure 3.1). The one-bit Equality function $EQ(x, y) = 1$ *iff* $x = y$. The one-bit AND function $AND(x, y) = 1$ *iff* $x = 1 \wedge y = 1$.

Figure 3.1: Deterministic Decision Tree for One-Bit Equality and One-Bit AND Function

We encourage the reader to see an excellent survey by Buhrman and de Wolf [BdW02] on the decision tree complexity of Boolean functions.

3.2.2 Randomized Query Model

A randomized decision tree $\mathcal{T}$ is simply a probability distribution μ on the deterministic decision trees $\{T_1, T_2, \ldots\}$ where the tree T_i occurs with probability $\mu(P_i)$. We say that $\mathcal{T}$ computes f correctly if for every input x: $\Pr_i[T_i(x) = f(x)] \geq 2/3$. The depth of $\mathcal{T}$ is the maximum depth of a T_i with $\mu(T_i) \neq 0$. *The (bounded-error) randomized query complexity* of f, denoted by $R(f)$, is

the minimum possible depth of a randomized tree computing f correctly on all inputs.

$$R(f) = \min_{\mu} \max_{T_i:\mu(T_i)\neq 0} depth(T_i).$$

Note that the above notion captures two sided bounded error query complexity models as the error can be both in 0 inputs as well as in 1 inputs. One can similarly define one sided error randomized query complexity. In this thesis we shall deal with two sided error model and thus we omit the definition of the latter.

3.2.3 Quantum Query Model

One can also define the quantum version of the decision tree model. Please refer to Chapter 2 for some basics of quantum computing.

Recall that unlike the classical input bit which can be in either 0 or 1 state, a quantum bit or qubit can be in a superposition of both 0 and 1 state.

$$\alpha_0|0\rangle + \alpha_1|1\rangle.$$

Here $\forall i, \alpha_i$ is a complex number and $\sum_i |\alpha_i|^2 = 1$.

In any classical query algorithm in a single query to the ith location of the input x, the value x_i is revealed. Formally, a query can be thought of as a transformation of a three register tuple (i,b,z). The query operation O reads the value of i and adds the value x_i to the second register b. It does not affect the workspace z.

$$O : (i, b, z) \rightarrow (i, b \oplus x_i, z).$$

Similarly, a single quantum query Q performs the following transformation.

$$Q : |i\rangle|b\rangle|z\rangle \rightarrow |i\rangle|b \oplus x_i\rangle|z\rangle.$$

Unlike the classical case, the registers are now quantum and can be in superposition of different states. Thus a single query can give us access to more than one location of the input x.

A quantum query algorithm starts with an N-qubit state $|0\rangle$ consisting of all zeros. Then it does two operations alternatively. Performs an unitary transformation U_0 to the quantum state. And makes a *quantum* query O. Depending on whether the ith bit of the basic state is zero or one,

the query negates the amplitude of each basic state.

A quantum algorithm with q queries looks like the following:

$$A = U_q O U_{q-1} \cdots O U_1 O U_0.$$

Here U_i's are the fixed unitary transformations independent of the input x. But the queries depend on the input x. At the end we measure the final state and produce output.

A bounded-error quantum query algorithm A computes f correctly if the final measurement gives the correct answer with probability at least $2/3$ for every input x. *The bounded-error quantum query complexity* of f, denoted by $Q(f)$, is the least q for which f admits a bounded-error quantum algorithm. We refer the reader to a survey by Buhrman and de Wolf [BdW02] for more literature on query complexity.

Deutsch's Algorithm

An example of a quantum query algorithm is Deutsch's Algorithm. Given a two-bit XOR function $XOR : \{0,1\}^2 \to \{0,1\}$ the function value is 1 iff only one of the two input bits is 1. It is easy to see that the deterministic query complexity of this function is 2, i.e., 2 queries are needed to compute the output of this function.

But Deutsch's quantum algorithm performs a single query to the input bits and finds out the answer with probability 1. The algorithm proceeds by performing Hadamard operations on two quantum registers which were set to $|01\rangle$ in the beginning. Then a single query to the input is performed. Lastly, after discarding the second quantum register and performing a Hadamard operation on the first register again we get the answer with certainty, i.e., we measure the first register and the answer is 1 iff the measurement result is 1. Formally, we start with $|0\rangle|1\rangle$.

$\rightarrow |0\rangle|1\rangle$

$\xrightarrow{H} \frac{1}{\sqrt{2}}\big(|0\rangle + |1\rangle\big)\frac{1}{\sqrt{2}}\big(|0\rangle - |1\rangle\big)$

$\rightarrow \frac{1}{2}|0\rangle\big(|0\rangle - |1\rangle\big) + \frac{1}{2}|1\rangle\big(|0\rangle - |1\rangle\big)$

$\xrightarrow{Q} \frac{1}{2}|0\rangle\big(|x_0\rangle - |1 \oplus x_0\rangle\big) + \frac{1}{2}|1\rangle\big(|x_1\rangle - |1 \oplus x_1\rangle\big)$

$\rightarrow \frac{1}{2}|0\rangle(-1)^{x_0}\big(|0\rangle - |1\rangle\big) + \frac{1}{2}|1\rangle(-1)^{x_1}\big(|0\rangle - |1\rangle\big)$

$\rightarrow \Big(\frac{1}{\sqrt{2}}\big((-1)^{x_0}|0\rangle + (-1)^{x_1}|1\rangle\big)\Big)\Big(\frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)\Big)$

Now by discarding the second register we have

$(-1)^{x_0}\frac{1}{\sqrt{2}}\big(|0\rangle + (-1)^{x_0 \oplus x_1}|1\rangle\big)$

$\xrightarrow{H} (-1)^{x_0}\big(x_0 \oplus x_1\big).$

So if we measure this register we will get $(x_0 \oplus x_1)$ as output.

We refer the reader to an excellent lecture notes by Watrous [Wat06] for the details of Deutsch's algorithm.

3.3 Query Complexity Measures

3.3.1 Some Classes of Boolean Function

Before we review some of the query complexity measures we first review some classes of Boolean functions.

3.3.2 Dual of a Function

The dual of a function f, denoted by f^*, is:

$$f^*(x) := \neg f(\neg x),$$

where $\neg x$ denotes the binary string obtained by flipping each bit in x.

Note that $f^{**} = f$ and that the deterministic, randomized and the quantum query complexity of a function f and its dual f^* are same.

Monotone Function

A Boolean function $f : \{0,1\}^n \rightarrow \{0,1\}$ is said to be *monotone* increasing if for any $x \leq y$, we have $f(x) \leq f(y)$, where $x \leq y$ means $x_i \leq y_i$ for all $i \in [n]$. Similarly, one can define a monotone decreasing function.

An example of a monotone-increasing function is the OR-function. $\text{OR}(x) = 1$ *iff* $\exists i \in [n]$ *s.t.* $x_i = 1$. Thus as soon as $x_i = 1$ for some $i \in [n]$ the OR-function becomes 1.

Note that if a function f is monotone, then so is f^*.

Transitive Function

A Boolean function $f(x_1, \ldots, x_n)$ is said to be *transitive* if there exists a group Γ that acts transitively on the variables x_is such that f is invariant under the group's action, i.e., for every $\sigma \in \Gamma$: $f(x_{\sigma_1}, \ldots, f_{\sigma_n}) = f(x_1, \ldots, x_n)$.

A simple example of a transitive function is a function whose output is 1 when there are two consecutive ones in the input bits. Formally, the function $f : \{0,1\}^n \to \{0,1\}$ is defined to be 1 *iff* $\exists i, j$ such that $j = i + 1 \ (mod\ n)$ *and* $x_i = x_j = 1$. This function is invariant under a cyclic permutation group.

Symmetric Function

A symmetric function is invariant under all permutation of its variables i.e., for all possible permutation σ: $f(x_{\sigma_1}, \ldots, f_{\sigma_n}) = f(x_1, \ldots, x_n)$.

A simple example of a symmetric function is the Majority function:
$MAJ(x_1, x_2, \cdots, x_n) = 1$ *iff* $\sum_i x_i \geq \frac{n}{2}$.

Evasive Function

A Boolean function $f : \{0,1\}^n \to \{0,1\}$ is said to be *evasive* if $D(f) = n$. So for an evasive function we cannot save even a single query.

Conjecture 3.3.1 (Evasiveness Conjecture [Lut01])**.** *Any non-constant monotone transitive function f on n variables has $D(f) = n$.*

Now we review some query complexity measures which we need for this thesis.

3.3.3 Sensitivity

The i^{th} bit of an input $x \in \{0,1\}^n$ is said to be sensitive for a function $f : \{0,1\}^n \to \{0,1\}$ if $f(x_1, \ldots, x_i, \ldots, x_n) \neq f(x_1, \ldots, 1 - x_i, \ldots, x_n)$. Note that $1 - x_i$ flips the value of x_i.

The sensitivity of f on x, denoted by $s_{f,x}$ is the total number of sensitive bits of the input x for f.

$$s_{f,x} = |\{i \in [n] \ : \ x_i \textit{ of } x \textit{ is sensitive for } f \ \}|.$$

The sensitivity of f, denoted by $s(f)$, is the maximum of $s_{f,x}$ over all possible choices of x.

$$s(f) = \max_x \ s_{f,x}.$$

The sensitivity of the all-zero input 0^n of OR-function is n, because for all-zero input flipping any single bit would change the function value to 1. Thus the sensitivity of the OR-function, $s(OR) = n$.

3.3.4 Block-sensitivity

A block $B \subseteq [n]$ of variables is said to be sensitive for f on input x, if flipping the values of all x_i such that $i \in B$ and keeping the remaining x_i the same, results in flipping the output of f.

The block sensitivity of f on an input x, denoted by $bs_{f,x}$ is the maximum number of *disjoint* sensitive blocks for f on x. The block sensitivity of a function f, denoted by $bs(f)$, is the maximum value of $bs_{f,x}$ over all possible choices of x.

$$bs(f) = \max_x \ bs_{f,x}.$$

Alternatively, the following integer program exactly captures the definition of block sensitivity of f on input x,

$$\begin{aligned} bs_{f,x} = \quad & \max \qquad \sum_{j=1}^{t} w_j \\ & \text{subject to} \quad \sum_{j:B_j \ni i} w_j \le 1, \qquad \forall i \in [n] \\ & \qquad\qquad\quad w_j \in \{0,1\}, \quad \forall j \in [t] \end{aligned} \tag{3.1}$$

The block sensitivity of f is computed by maximizing $bs_{f,x}$ over all possible inputs $x \in \{0,1\}^n$.

It is known that $D(f) \ge R(f) \ge bs(f) \ge s(f)$. For monotone functions, $bs(f) = s(f)$.

Fractional Block Sensitivity

Now by relaxing the above integer program we get something called the fractional block sensitivity.

$$
\begin{aligned}
fbs_{f,x} = \quad & \max \quad && \sum_{j=1}^{t} w_j \\
& \text{subject to} && \sum_{j:B_j \ni i} w_j \le 1, \qquad \forall i \in [n] \\
& && w_j \in [0,1], \quad \forall j \in [t]
\end{aligned} \tag{3.2}
$$

The fractional block sensitivity of f is computed by maximizing $fbs_{f,x}$ over all possible inputs $x \in \{0,1\}^n$.

3.3.5 Certificate Complexity

A partial assignment $A : [n] \to \{0,1,*\}$ of a Boolean function $f : \{0,1\}^n \to \{0,1\}$ partially assigns some values to the input x of the function f. Let $|A|$ denote the number of values that the function A assigns to 0 or 1, and A is consistent with an input x if and only if $A(i) = x_i$ or $A(i) = *$, for all i.

Definition 3.3.2. *For $b \in \{0,1\}$, a partial assignment A is a b-certificate for function f for all x if $f|_A(x) = b$.*

For $b \in \{0,1\}$, the b-certificate complexity of f is defined by

$$C_b(f) = \max_{x:f(x)=b} \min_{A} \{|A| : f|_A(x) = b\}.$$

The certificate complexity of a function f is the maximum of $C_1(f)$ and $C_0(f)$.

For example $C_1(AND) = n, C_0(AND) = 1$ and thus $C(AND) = n$.

All these measures, sensitivity $s(f)$, block sensitivity $bs(f)$ and certificate complexity $C(f)$ are related. It is easy to see that $s(f) \le bs(f) \le C(f)$.

Furthermore it has long been known that,

Lemma 3.3.3 (Gilmer et al. [GSS13])**.** *For all total functions f,*

$$C(f) \le s(f)bs(f) \le bs(f)^2.$$

For Chapter 5 we also require the definition of a *minimal certificate* of a function f.

3.3.6 Minimal Certificate

A *minimal certificate* of size s for a monotone increasing function f is an input z such that (a) The hamming weight of z, i.e., $|z|$, is s, (b) $f(z) = 1$, and (c) for any y with $|y| < s$, $f(y) = 0$. Every minimal certificate z can be uniquely associated with the subset $S_z := \{i \mid z_i = 1\}$.

We shall use the minimal certificate size to give lower bounds to the quantum query complexity of function f.

Lemma 3.3.4 (Minimal Certificate [BdW02])**.** *If f has a minimal certificate of size s then the quantum query complexity of f,*

$$Q(f) = \Omega(\sqrt{s}).$$

3.3.7 Degree of a Function

Before we go into the definition of *degree of a function* we first review multi-linear polynomials.

Definition 3.3.5. *Multi-linear Polynomials: A multi-linear polynomial p is a polynomial that is linear in each of its variables i.e., each monomial is a constant times a product of distinct variables. The degree of a multi-linear polynomial is the maximum over the number of distinct variables in any monomial.*

For any variable x_i of a Boolean function f, $x_i = x_i^k$ for any k. Thus in a single monomial, of any polynomial representing a Boolean function, variables do not need to occur more than once.

We need the next well known lemma for the definition of the degree of a function. It says that if two multi-linear polynomials p and q have same output on every single inputs and they have degrees at most d, then they are identical to each other (see [BdW02] for the proof).

Lemma 3.3.6. *Let $p, q : \mathbb{R}^n \to \mathbb{R}$ be two multi-linear polynomials of degree upper bounded by d. If $p(x) = q(x)$ for all inputs $x \in \{0,1\}^n$ with $|x| \leq d$, then these two polynomials are same, $p = q$.*

Now we define the degree of a function. A function f is represented by a polynomial $p : \mathbb{R}^n \to \mathbb{R}$ if $f(x) = p(x)$ for all inputs x. Since for Boolean algebra, $x^k = x, \forall k \geq 1$ we can assume that the representing function is a multi-linear function. Furthermore, because of the previous lemma, Lemma 3.3.6, the representing polynomial is unique.

$$f(x_1, x_2, \cdots, x_n) = \sum_{S \subseteq [n]} \alpha_S \prod_{i \in S} x_i.$$

Definition 3.3.7. *Degree of a function: The degree of a Boolean function f is the degree of the unique multi-linear polynomial p that represents f.*

For example, the representing polynomial of the AND-function is $\text{AND}(x_1, x_2, \cdots, x_n) = \prod_{i=1}^{n} x_i$. The polynomial is the monomial $x_1 x_2 \cdots x_n$ itself. And thus the degree of the AND-function is n.

One might wonder if most Boolean functions have full degrees. Shi and Yao (unpublished) gave the following lemma which exactly characterises the condition of having full degree (see [BdW02] for the proof).

Lemma 3.3.8. *$deg(f) = n$ if and only if the number of inputs x for which $f(x) = 1$ and $|x|$ is even, is not equal to the number of inputs x for which $f(x) = 1$ and $|x|$ is odd. Here $|x|$ is the hamming weight of the input x.*

It can be calculated that almost all the functions (except a constant fraction of all possible functions) satisfy the condition of the Lemma 3.3.8 and thus most of the functions have full degrees [BdW02].

Degree of a function can be significantly lower than the sensitivity and block sensitivity of that function. Meanwhile it can also be significantly larger than sensitivity, block sensitivity and the certificate complexity.

In particular the following two lemmas highlights the relationship between block sensitivity and degree of a function f.

Lemma 3.3.9 (Nisan & Szegedy [NS94])**.** *For all total functions f,*

$$bs(f) \leq 2deg(f)^2.$$

Lemma 3.3.10 (Gilmer et al. [GSS13])**.** *For all total functions f,*

$$deg(f) = \tilde{O}(bs(f)^3).$$

Here the $\tilde{O}(\cdot)$ notation hides a $poly(\log bs(f))$ factor. The AND-OR function $(\bigwedge_{i\in[m]} \bigvee_{j\in[n]} x_{ij})$ achieves the following

$$\exists \text{ a total function } f, deg(f) = \Omega(bs(f)^2).$$

3.3.8 Approximate Degree of a Function

Sometime, instead of finding the exact polynomial p that represents f, we might want to find another function f' whose representing polynomial p' has lower degree than that of the polynomial p and f' approximates the function f.

Let $\mathcal{F}_n = \{f : \{0,1\}^n \to \{0,1\}\}$ is the set of all Boolean functions on n variables.

Definition 3.3.11 (Approximate Degree:)**.** *Let $f \in \mathcal{F}_n$ and $\epsilon \in (0, 1/2)$. The ϵ-approximate degree of f, denoted by $\widetilde{deg}_\epsilon(f)$, is the smallest $d \geq 1$ such that there exist another function $f' \in \mathcal{F}_n$ with $deg(f') = d$ and f' approximates f, i.e., $|f'(x) - f(x)| \leq \epsilon$, for all inputs $x \in \{0,1\}^n$.*

Approximate degree can be significantly lower than the degree of the function.

For example, the OR-function has $deg(OR) = \Omega(n)$ whereas $\widetilde{deg}(OR) = O(\sqrt{n})$.

While $deg(OR) = \Omega(n)$ comes from the fact that OR-function is a symmetric function and for all symmetric functions f, $deg(f) \geq s(f) = \Omega(n)$ [See Lemma 3.6.2], $\widetilde{deg}(OR) = O(\sqrt{n})$ can be achieved by using Chebyshev polynomials [See Example 3.11 of [NS94]].

It turns out that if a function f has high sensitivity or high block sensitivity then it also has high approximate degree.

Lemma 3.3.12 (Nisan & Szegedy [NS94])**.** *for all total functions f,*

$$i)\ \widetilde{deg}(f) = \Omega(\sqrt{s(f)}),$$

$$ii)\ \widetilde{deg}(f) = \Omega(\sqrt{bs(f)}).$$

The OR-function is an example where both bounds of these lemmas are asymptotically tight. The approximate degree is known to be upper bounded by $bs(f)^3$, i.e., for all total functions f,

$$\widetilde{deg}(f) = O(bs(f)^3).$$

It is unknown if the same kind of relationship holds between degree of a function and the sensitivity of the function.

Open 3.3.13. *For all total functions f,*

$$\widetilde{\deg}(f) = O(s(f)^{O(1)})?$$

Resolving this would resolve the following long-standing *Sensitivity Conjecture*.

Conjecture 3.3.14 (Sensitivity Conjecture, Nisan and Szegedy [NS94])**.** *For all total functions f,*

$$bs(f) \leq poly(s(f)).$$

It turns out that the two measures, degree of a function, $deg(f)$ and the approximate degree of a function, $\widetilde{deg}(f)$ are polynomially related.

Lemma 3.3.15 (Nisan & Szegedy; Beals et al. [NS94, BBC+01])**.** *For all total functions f,*

$$deg(f) = O(\widetilde{deg}(f)^6),$$

and for the AND-function the following holds

$$\exists f,\ deg(f) = \Omega(\widetilde{deg}(f)^2).$$

Thus the exact relationship of degree and approximate degree is yet to be known. Readers may look into a nice survey on the approximate degree of Boolean functions by Shi [Shi07].

3.4 Relationship of Query Complexity and Different Measures

In this section we review the relationship between different complexity measures and different query complexity models. With a notable exception of the sensitivity, all the complexity measures such as block sensitivity, degree, approximate degree, certificate complexity etc. and the complexity classes such as $D(f), R(f), Q(f)$ are all known to be polynomially related.

3.4.1 Deterministic Query Complexity

Deterministic query complexity $D(f)$ of a function f is lower bounded by $bs(f)$ and $deg(f)$. It turns out that it is also upper bounded by $bs(f)^3$ [BdW02].

For all total functions f,

$$D(f) \leq C^1(f)bs(f) \leq C(f)bs(f) \leq s(f)bs(f)^2 \leq bs(f)^3.$$

One of the important open problems is to know if the following is true.

Open 3.4.1. *For all total functions f,*

$$D(f) \leq bs(f)^2?$$

Solving this open problem would solve several other open problems in query complexity. For more details of such consequences please look into the Page 5 of [ABDK15].

Deterministic query complexity is also upper bounded in terms of degree and approximate degree. For all total functions f,

$$i)\ D(f) = O(deg(f)^3),$$

$$ii)\ D(f) \leq 6\widetilde{deg}(f)^6.$$

The above upper bounds are not known to be tight. It has been shown that there exists a total Boolean f such that $D(f) = \Omega(deg(f)^2)$ [GPW15] and there exist another total Boolean function f' for which $D(f') = \Omega(\widetilde{deg}(f')^4)$ [ABB+15].

3.4.2 Randomized Query Complexity

Two-sided error randomized query complexity, $R(f)$ is lower bounded by block sensitivity and approximate degree. For all total functions f,

$$i)\ \widetilde{deg}(f) \leq R(f),$$

$$ii)\ bs(f) \leq 3R(f).$$

Deterministic query complexity, $D(f)$, is upper bounded by $O(R(f)^3)$ [ABB+15]. In terms of upper bounds randomized query complexity has the following relationship with degree and approximate degree: for all total functions $f, R(f) = O(deg(f)^3)$ and $\exists$ *a total function* f', such that $R(f') = \Omega(deg(f')^2)$ [GJPW15] and for all total functions $f, R(f) = O(\widetilde{deg}(f)^6)$ and there exists another total function f'', such that $R(f'') = \Omega(\widetilde{deg}(f'')^4)$ [ABB+15]. Resolving the open question Open 3.4.1 would make the lower bound tight.

3.4.3 Quantum Query Complexity

Quantum Query complexity $Q(f)$ of a function f is lower bounded by the approximate degree $\widetilde{deg}(f)$ of the function f. For all total functions f,

$$\widetilde{deg}(f) \leq 2Q(f).$$

An upper bound in terms of approximate degree is known, $Q(f) = O(\widetilde{deg}(f)^6)$. Very recently Aaronson et al. [ABDK15] have shown that there exists a total function f which achieves a forth-power gap between its quantum query complexity and approximate degree, i.e., $Q(f) = \Omega(\widetilde{deg}(f)^4)$. Note that again resolving the open question Open 3.4.1 would make the above bound tight.

Although $D(f), R(f)$ and $Q(f)$ are polynomially related the exact relationship between them is still not known. For instance it is known that $R(f) \leq D(f) = O(Q(f)^6)$. On the other hand, in the recent paper Aaronson et al. [ABDK15] have shown that there exist a total function f for which $R(f) = \Omega(Q(f)^{2.5})$. Ambainis et al. [ABB+15] also have demonstrated another total function f for which $D(f) = \Omega(Q(f)^4)$. Yet again note that resolving the open question Open 3.4.1 would tighten the above relations.

3.5 Query Complexity for Monotone Functions

For monotone functions, query complexity is much more well understood. The sensitivity conjecture (see Conjecture 3.3.14) is true for monotone functions. In fact, for monotone functions sensitivity, block sensitivity, and certificate complexity all are equivalent.

For all total monotone functions f, $s(f) = bs(f) = C(f)$.

And hence, $D(f) \leq s(f)^2$ *and* $s(f) \leq deg(f)$.

Also, $D(f) = O(R(f)^2)$ *and* $D(f) = O(Q(f)^4)$.

3.6 Query Complexity for Symmetric Functions

Recall that a symmetric function is invariant under all permutation of its variables.

Lemma 3.6.1 (Paturi [Pat92]). *Let f be a function on n variables such that $f(z) = 0$ for all z with $|z| = t - 1$ and $f(z) = 1$ for all z with $|z| = t$. Then: $\widetilde{\deg}(f) = \Omega(\sqrt{t(n-t)})$.*

Lemma 3.6.2 (Turán [Tur84]). *For all total symmetric functions f,*

$$s(f) \geq \lceil \frac{n+1}{2} \rceil.$$

Chapter 4

Communication Complexity

This chapter briefly reviews different types of communication complexity models. In Section 4.3 we define Merlin-Arthur protocols which is a communication complexity analogue of the Turing machine class MA.

4.1 Communication Complexity Models

In the communication complexity model two players Alice and Bob are given two inputs x and y. Their goal is to compute a predefined function $f(x, y)$. As (usually) the function depends on both x and y and as Alice and Bob both only know their respective private inputs, they need to communicate in order to compute the function. Typically we are interested in the minimum amount of communication to accomplish the job.

The most general model is the two-way communication complexity model where both Alice and Bob send message to each other. Another well-studied model is the one-way communication complexity where only either Alice or Bob is allowed to send messages to the other player. The message receiving player computes the output based on his/her input and the messages received. Depending on the different resources used for the communication, there are several variants of communication complexity models such as randomized or quantum communication complexity where the players are allowed to toss coins (jointly or privately) or use quantum resources. We shall briefly review all these models in this chapter.

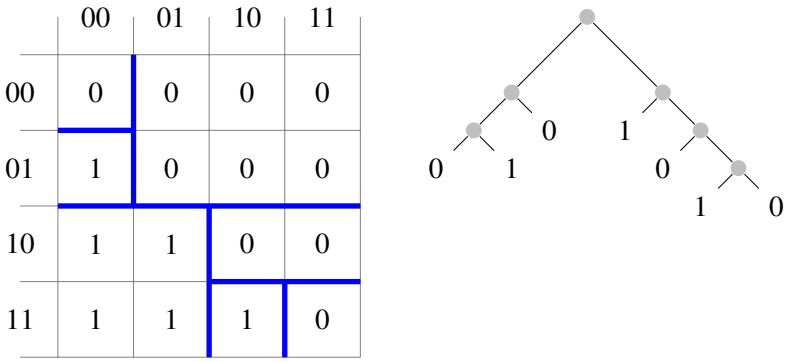


Figure 4.1: Communication Matrix and its Corresponding Protocol tree

4.1.1 Deterministic (Two-way) Communication Complexity

Definition 4.1.1. *Let $f : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ be a Boolean function. In a communication complexity protocol two players Alice and Bob receive inputs x and y from $\{0,1\}^n$ respectively. Their goal is to compute the function $f(x,y)$. In the protocol players exchange messages in order to compute $f(x,y)$. Such a protocol is represented by a binary tree (a.k.a the protocol tree), in which vertices, alternating by layer, belong to Alice or to Bob, edges are labelled with messages, and leaves either accept or reject. The value of the protocol on an input (x,y) is the value of the corresponding leaf that is reached by traversing the tree, starting from the root of the tree.*

We say a protocol P correctly computes the function $f(x,y)$ if for all x, y the output of the protocol $P(x,y)$ is equal to $f(x,y)$. The communication complexity of a protocol is the maximum number of bits exchanged for all x, y i.e., the maximum length of the protocol tree.

The communication matrix M_f is a $2^n \times 2^n$ matrix where the rows are labelled by all x and the columns are labelled by all y. The value of any (x,y)th entry of the matrix is the value $f(x,y)$.

Any protocol P computing a function f induces a partition of the communication matrix M_f into monochromatic rectangles (all the entries are either all 0s or all 1s). The number of induced monochromatic rectangles is the number of leaves in the protocol tree of P. Figure 4.1 illustrates a communication matrix partition and its corresponding protocol tree.

The deterministic communication complexity $D(f)$ of a function f is the complexity of a best protocol that computes f for all (x,y).

Every valid protocol for computing a function f yields a valid protocol tree where each leaf corresponds to a monochromatic partition. And for each (x,y) there is a path from the root of the tree to some leaf of the tree. Thus every valid protocol yields a monochromatic partition of its communication matrix M_f. The number of monochromatic rectangles can be used to give lower

	00	01	10	11
00	0	0	0	0
01	0	1	0	0
10	0	0	0	0
11	0	0	0	0

Figure 4.2: Partition Does Not Correspond to Any Valid Protocol

bounds to the deterministic communication complexity of the function f.

Lemma 4.1.2. *If any partition of the communication matrix M_f requires t monochromatic rectangles, then $D(f) \geq \log t$.*

However, note that not all partition, of the matrix into monochromatic rectangles, corresponds to a valid communication protocol. See Figure 4.2 for an example. Although it is a valid partition of the matrix into several monochromatic sub-matrices, it does not correspond to any protocol. Any valid partition corresponding to a protocol must have the following property: Given the whole rectangle it is divided into exactly two halves (two disjoint set of rows). And then these two halves are recursively partitioned into another two halves (now two disjoint set of columns). This process continues until we reach a point where all partitions are monochromatic. Notice that for the above example (Figure 4.2) this is not possible.

In fact very recently a nearly quadratic separation between the deterministic communication complexity and the logarithm of the partition number has been shown by Ambainis et al. [AKK16]. So the above lemma is not tight.

Furthermore, Aho et al. [AUY83] have shown that for all Boolean function f, $D(f) \leq O(\log^2 t)$. Thus the result by Ambainis et al. [AKK16] is essentially tight.

Fooling Set

Fooling set is one of the fundamental technique to give lower bound to the deterministic communication complexity. Let $\mathcal{X}$ and $\mathcal{Y}$ be the set of all possible inputs x of Alice and all possible inputs y of Bob respectively. Then the fooling set is defined as follows:

Definition 4.1.3. *Fooling Set: A subset $S \subseteq \mathcal{X} \times \mathcal{Y}$ is called a fooling set if for some $b \in \{0,1\}$ and for all $(x,y) \in S, f(x,y) = b$ and for any two (x,y) and $(x',y') \in S$ either $f(x,y') \neq b$ or*

$f(x', y) \neq b$.

Lemma 4.1.4. *For any function $f(x, y)$, $D(f) \geq \log |S|$. Here S is the fooling set of the function f.*

For example, the Equality function can be lower bounded by the fooling set method. Equality is defined as: $EQ(x, y) = 1$ *iff* $x_i = y_i \forall i \in [n]$.

Now the fooling set S of EQ is: $S = \{(x, y) \mid x = y\}$. Thus $|S| = 2^n$ and $D(EQ) \geq n$.

4.1.2 Non-deterministic Communication Complexity

Definition 4.1.5. *The non-deterministic communication complexity $N(f)$ of a Boolean function f is the length of the communication in an optimal two-player protocol in which Alice and Bob can make non-deterministic guesses, and there are two possible outputs* `accept`, `reject`. *For each x, y with $f(x, y) = 1$ there is a guess that will make the players accept but there is no guess that will make the players reject, and vice versa for inputs with $f(x, y) = 0$.*

Note that the above is the two-sided version of non-deterministic communication complexity. It is well known [KN97] that $N(f) \leq D(f) \leq O(N^2(f))$, and that these inequalities are tight.

4.1.3 Randomized Communication Complexity

Definition 4.1.6. *In a public coin randomized protocol for f the players have access to a public source of random bits. And we are required that for all inputs x, y, the randomized protocol gives the correct output with probability $1 - \epsilon$ for some $\epsilon < 1/2$.*

A randomized protocol can be though of as a probability distribution μ over the deterministic protocols $\{P_1, P_2, \cdots\}$, where each deterministic protocol P_i occurs with probability $\mu(P_i)$ We say that a randomized protocol computes f correctly if for every input x: $\Pr_i[P_i(x) = f(x)] \geq 1 - \epsilon$. The running time of the randomized protocol is the maximum depth of the protocol tree of a protocol P_i with $\mu(P_i) \neq 0$.

The (bounded-error) randomized communication complexity *of f, denoted by $R(f)$, is the minimum possible running time of a randomized protocol computing f correctly on all inputs.*

$$R(f) = \min_{\mu} \max_{P_i : \mu(P_i) \neq 0} depth(P_i).$$

Note that the above defined model is a two-sided error model i.e., the error can be both in 1-inputs and 0-inputs. Similarly, a one-sided error model can also be defined.

Lemma 4.1.7 (Min-Max Lemma (Yao)). *Let μ be a probability distribution on the* inputs $\mathbb{X} \times \mathbb{Y}$. *Then the distributional complexity of f, $D_{\mu,\epsilon}(f)$ is the cost of the best deterministic protocol P which gives correct answer on $(1-\epsilon)$ fraction of all the inputs $\mathbb{X} \times \mathbb{Y}$. Here the fraction is calculated based on the distribution μ.*

Then,

$$R_{\epsilon}^{pub}(f) = \max_{\mu} D_{\mu,\epsilon}(f).$$

This lemma says that in order to give a lower bound to the public coin randomized communication complexity of any function it suffices to find a hard distribution μ on the input and then give a lower bound on the deterministic cost $D_{\mu,\epsilon}(f)$.

Private coin protocols are defined analogously (players now have access only to private random bits), and their complexity is denoted by $R_{\epsilon}(f)$. It can be shown that with an additive $\log n$ bits (when the input size in n) any private coin protocol for computing f can be converted into a public coin protocol computing the same function f [KN97].

4.1.4 Quantum Communication Complexity (With Entanglement)

In this model the players are quantum and they are allowed to send quantum messages between each other. Both players Alice and Bob have their private set of qubits. Before the protocol starts some of the qubits are initialized to the their respective inputs. Other qubits are kept in $|0\rangle$ state. They may also have some shared entangled states. In each round of the protocol the players alternatively perform a unitary operation on the qubits in his/her possession and then send one of his/her qubits to the other player. Note that the players choose in advance which qubits to send and which unitary operation to perform in each round. At the end of the protocol one of the player's some of the qubits are measured and the output is produced. The complexity of the protocol is the number of qubits exchanged.

As usual we are interested in the minimum number of qubits exchanged to compute a function f. The exact quantum communication complexity $QE(f)$ of a function f is the complexity of an optimal quantum protocol that computes f without any error.

We also define a bounded-error version of the quantum communication complexity denote by $Q(f)$, where the protocol is allowed to have error, but for all inputs (x, y) the protocol has to be correct with probability $1-\epsilon$, for $\epsilon < 1/2$. When we allow shared entanglement we denote the same by $Q^*(f)$.

Bounded error quantum communication complexity can be significantly smaller than the randomized one [BCW98]. For example the Disjointness function ($DISJ(x,y) = 1$ *iff* $\nexists i \in [n]$ *s.t.* $x_i = y_i = 1$) has quantum bounded error communication complexity $\Theta(\sqrt{n})$, whereas the randomized communication complexity is $\Omega(n)$.

Partial Functions

Definition 4.1.8. *Partial Functions: A partial Boolean function f is a function which is not defined for all the inputs. The function $f : \{0,1\}^n \times \{0,1\}^m \times \{0,1,\perp\}$ has three possible outputs, $\{0,1,\perp\}$, where $\perp$ stands for "undefined" and the output is $\perp$ for the inputs which are not defined.*

A protocol for f is correct, if it gives the correct output for all x, y with $f(x,y) \neq \perp$ (with certainty for deterministic protocols, and with probability 2/3 for randomized and quantum protocols).

4.2 One-way Communication Complexity

One-way Protocols

A protocol is one-way, if Alice sends a message to Bob, who computes the function value, or vice versa. We denote by $D^{A\to B}(f)$ the deterministic one-way communication complexity of a function f, when Alice sends the message to Bob. Similarly, the randomized complexity is defined by $R^{A\to B}(f)$.

When considering an one-way protocol from Alice to Bob, in order to reduce the message size Alice needs to map more than one inputs into a single message. Two rows x, x' of A_f are *distinct*, if there is a column y, such that $f(x,y) = 1$ and $f(x',y) = 0$ or vice versa, i.e., the function values differ on some defined input.

Similar to the above, we denote by $Q^{A\to B}(f)$ the quantum one-way communication complexity of f with error 1/3. This notion is of course asymptotically robust when it comes to changing the error to any other constant. $Q^{A\to B,*}(f)$ denotes the complexity if Alice and Bob share entanglement.

4.3 Merlin Arthur Protocols

We now define some more esoteric modes of communication that extend the standard nondeterministic mode to the quantum case. We restrict our attention to one-way protocols.

One-way Merlin Arthur Protocols

Definition 4.3.1. *In a one-way MA-protocol there are 3 players Merlin, Alice, Bob. Merlin sends a classical message to Alice, who sends a classical message to Bob, who gives the output. Alice and Bob share a public coin, which is not seen by Merlin. For a Boolean function* $f : \{0,1\}^n \times \{0,1\}^m \to \{0,1\}$ *the protocol is correct, if for all 1-inputs there is a message from Merlin, such that with probability 2/3 Bob will accept, whereas for all 0-inputs, and all messages from Merlin, Bob will reject with probability 2/3. The communication complexity is defined as usual and denoted by* $MA^{A\to B}(f)$.

A one-way QCMA-protocol is defined similarly, but whereas Merlin's message is still classical, Alice can send a quantum message to Bob, and Alice and Bob may share entanglement. The complexity with shared entanglement is denoted $QCMA^{A\to B,*}(f)$.

In a one-way QMA-protocol also Merlin's message may be quantum. The complexity is denoted by $QMA^{A\to B}(f)$ *in the case where no entanglement is allowed.*

4.4 k-Round Communication Complexity

Definition 4.4.1. *The deterministic communication complexity of protocols with at most* k *messages exchanged, starting with Alice, is denoted by* $D_k(f)$.

4.5 Simultaneous Message Passing (SMP) Model

Definition 4.5.1. *In a simultaneous message passing protocol, both Alice and Bob send messages* m_A, m_B *to a referee. The referee, based on* m_A, m_B*, computes the output. The simultaneous communication complexity of a function* f*,* $R^{\|}(f)$*, is the cost of the best SMP protocol that computes the function* f *using private randomness and error 1/3.*

For more details about classical communication complexity we encourage the reader to read the excellent monography by Kushilevitz and Nisan [KN97] and for quantum communication complexity readers can follow an excellent survey by de Wolf [dW02].

Chapter 5

Query Complexity of Graph Properties

This chapter is based on the following paper:

- *Kulkarni, R., and Podder, S. (2016). Quantum Query Complexity of Subgraph Isomorphism and Homomorphism. In Proceedings of the 33rd Symposium on Theoretical Aspects of Computer Science (STACS 2016), Pages 48:1–48:13 [KP16]*

5.1 Subgraph Isomorphism and Homomorphism Problem

Understanding the query complexity of monotone graph properties has a long history. In the deterministic setting the Aanderaa-Rosenberg-Karp Conjecture asserts that one must query all $\binom{n}{2}$ edges in the worst-case. The randomized complexity of monotone graph properties is conjectured to be $\Omega(n^2)$. Yao [Yao87] obtained the first super-linear lower bound in the randomized setting using graph packing arguments. Subsequently his bound was improved by King [Kin88] and later by Hajnal [Haj91]. The current best known bound is $\Omega(n^{4/3}\sqrt{\log n})$ due to Chakrabarti and Khot [CK01]. Moreover, O'Donnell, Saks, Schramm, and Servedio [OSSS05] also obtained an $\Omega(n^{4/3})$ bound via a more generic approach for monotone transitive functions. Friedgut, Kahn, and Wigderson [FKW02a] obtain an $\Omega(n/p)$ bound where the p is the critical probability of the property. In the quantum setting, Buhrman, Cleve, de Wolf and Zalka [BCdWZ99] were the first to study quantum complexity of graph properties. Santha and Yao [SY] obtain an $\Omega(n^{2/3})$ bound for general monotone properties. Their proof follows along the lines of Hajnal's proof.

In this chapter we focus on two of the most interesting problems among the monotone graph properties – the Subgraph Isomorphism Problem and the Subgraph Homomorphism Problem. Below we formally define the Subgraph Isomorphism Problem and prove the related results.

We shall discuss the Subgraph Homomorphism Problem in Section 5.3 of this chapter. To briefly define the problem first recall that a homomorphism from a graph H into a graph G is a function $h : V(H) \to V(G)$ such that: if $(u, v) \in E(H)$ then $(h(u), h(v)) \in E(G)$. Then given a graph G the Subgraph Homomorphism Problem asks whether a graph H admits a homomorphism into G.

5.2 Subgraph Isomorphism Problem

Let H be a (non-empty) graph on n vertices, possibly containing isolated vertices and let G be an unknown input graph (on n vertices) given by query access to its edges, i.e., queries of the form "Is $\{i, j\}$ an edge in G?". We say $H \leq G$ if G contains H as a (not necessarily induced) subgraph. Let $f_H : \{0,1\}^{\binom{n}{2}} \to \{0,1\}$ be defined as follows:

$$f_H(G) = \begin{cases} 1 & \text{if } H \leq G \\ 0 & \text{otherwise.} \end{cases} \tag{5.1}$$

The well-known Graph Isomorphism Problem asks whether a graph H is isomorphic to another graph G. It is not known to be solvable in polynomial time. It is also highly unlikely that it will be a NP-complete problem. Thus the problem may be in the complexity class NP-intermediate. Very recently a quasi-polynomial time algorithm for Graph Isomorphism problem has been proposed by Babai [Bab15].

The Subgraph Isomorphism Problem is a generalization of the Graph Isomorphism Problem where one asks whether H is isomorphic to a *subgraph* of G. Unlike the graph isomorphism problem this problem is NP-Complete. Several central computational problems for graphs such as containing a clique, containing a Hamiltonian cycle, containing a perfect matching can be formulated as instances of the Subgraph Isomorphism Problem by fixing H appropriately. Given the generality and importance of the problem, people have investigated various restricted special cases of this problem in different models of computation [wik, LRR14]. In the context of query complexity, in 1992 Dietmar [Die92] studied this problem in the randomized setting and showed that $R(f_H) = \Omega(n^{3/2})$, which is the best known bound to this date. In this chapter we investigate this problem in the quantum setting.

To the best of our knowledge, the quantum query complexity for the Subgraph Isomorphism Problem has not been investigated prior to our work when H is allowed to be any graph on n

vertices. A special case of this problem when H is of a constant size has been investigated before for obtaining upper bounds [LMS11].

5.2.1 Related Work

Dietmar [Die92] obtained an $\Omega(n^{3/2})$ bound for the randomized query complexity of the Subgraph Isomorphism. This is currently the best known bound for the Subgraph Isomorphism Problem. Until very recently[1], it was believed that for all total functions the quantum query complexity is at least the square root of the randomized one. In this work we address the quantum query complexity of the Subgraph Isomorphism Problem and obtain the square root of the current best randomized bound.

The main difference between the previous work and this one is that all the previous works, including that of Santha and Yao [SY], obtained the lower bounds based on an embedding of a *tribes* function [BBC+01] on a large number of variables in monotone graph properties[2]. Recall that the tribes function with parameters k and ℓ, is a function $T(k,\ell)$ on $k \cdot \ell$ variables defined as: $\bigvee_{i\in[k]} \bigwedge_{j\in[\ell]} x_{ij}$. This method yields a lower bound of $\Omega(k\cdot\ell)$ for the randomized query complexity and $\Omega(\sqrt{k\cdot\ell})$ for quantum. We deviate from this line by embedding a threshold function T_n^t instead of a tribes. Recall that $T_n^t(z_1,\ldots,z_n)$ is a function on n variables that evaluates to 1 if and only if at least t of the z_i's are 1. Since the randomized complexity of T_n^t is $\Theta(n)$, this does not give us any advantage for obtaining super-linear *randomized* lower bounds. However, it *does* yield an advantage for the quantum lower bounds as the quantum query complexity of T_n^t is $\Theta(\sqrt{t(n-t)})$ [Pat92], which can reach up to $\Omega(n)$ for $t = \Theta(n)$. Since this technique works only in the quantum setting, the randomized versions of our bounds remain intriguingly open.

We now prove the quantum query complexity lower bound for the Subgraph Isomorphism Problem.

5.2.2 Lower Bound on the Subgraph Isomorphism Problem

This section gives a lower bound on the quantum query complexity of the Subgraph Isomorphism Problem for H in terms of the maximum independence number of H.

[1] Very recently this has been falsified by Aaronson et al. [ABDK15].
[2] Similar tribes-embeddings were used for obtaining lower bounds for matroidal Boolean functions in [KS13].

Theorem 5.2.1. *For any non-empty H,*

$$Q(f_H) = \Omega\left(\sqrt{\alpha_H \cdot n}\right),$$

where α_H denotes the size of a maximum independent set of H on n vertices. For example when H is a single edge, $\alpha_H = n - 1$ (the independent set consists of one vertex from the edge and rest of the $n - 2$ isolated vertices).

The proofs crucially rely on the duality of monotone functions and appropriate embeddings of tribes and threshold functions. All the lower bounds also hold for the approximate degree $\widetilde{\deg}(f)$, which is a lower bound on the quantum query complexity [BBC+01, Amb03].

Before proving Theorem 5.2.1 we first prove two lemmas.

Let S_d denote the star graph with d edges. Then f_{S_d} is the property of having a vertex of at least degree d. First we show:

Lemma 5.2.2. *For $1 \le d \le n - 1$,*

$$Q(f_{S_d}) = \Omega(n).$$

Proof. We divide the proof into two cases:

Case 1: $d > n/2$.

Fix a clique on the vertices $1, \ldots, \lfloor n/2 \rfloor$ and fix an independent set on the vertices $\lfloor n/2 \rfloor + 1, \ldots, n$. Note that we still have $\lfloor n/2 \rfloor \times \lceil n/2 \rceil$ edge-variables that are not yet fixed. Now as soon as any vertex v from the clique has $(d - \lfloor n/2 \rfloor + 1)$ edges to the independent set present, we have a d-star. Thus f_{S_d} becomes an $OR_{\lfloor n/2 \rfloor} \circ T_{\lceil n/2 \rceil}^{(d-\lfloor \frac{n}{2} \rfloor + 1)}$ function, which has a lower bound of $\Omega(n)$ via the Composition Theorem for quantum query complexity [LMR+11].

Case 2: $d \le n/2$.

A minimum certificate of f_{S_d} is a d-star.

Now we have the following lemma due to Yao [Yao87].

Lemma 5.2.3 (Packing Lemma [Yao87])**.** *If z_1 is a minimal certificate of $\mathcal{P}$ and z_2 is a minimal certificate of $\mathcal{P}^*$ then z_1 and z_2* cannot *be packed together.*

By the above Lemma 5.2.3 we know that this d-star cannot be packed with any minimal certificate of the dual $f^*_{S_d}$. Thus every vertex in the dual $f^*_{S_d}$ must have degree $> n - d$. Hence the minimal certificate size is at least $\Omega(n^2)$ and $Q(f^*_{S_d}) = Q(f_{S_d}) = \Omega(n)$.

□

Recall that a *Threshold function* $T_n^t(z_1, \ldots, z_n)$ is a function on n variables such that T_n^t outputs 1 if and only if at least t variables are 1.

Let t denote the smallest integer such that $f_H^*(K_t) = 1$, where K_t denotes the complete graph on t vertices. Note that $t = \alpha_H + 1$.

Lemma 5.2.4.

$$Q(f_H) \geq \Omega(\sqrt{t(n-t)}).$$

Proof. We *embed* T_n^t in f_H^* (on inputs of Hamming weight $t-1$ and t) via the following mapping: Let $x_{ij} := z_i \cdot z_j$ and let $f'(z_1, \ldots, z_n) := f_H^*(\{x_{ij}\})$. Note that $f' \equiv T_n^t$. Also note[3] that $Q(f_H) = Q(f_H^*)$ and $\widetilde{deg}(f') \leq 2 \cdot \widetilde{deg}(f_H^*)$. Since $Q(f) \geq \widetilde{deg}(f)$, it remains to prove the following:

Claim 5.2.5. $\widetilde{deg}(f') = \Omega(\sqrt{t(n-t)})$

We need the Lemma 3.6.1 due to Paturi [Pat92]: Note that f' ($\equiv T_n^t$) satisfies the condition of the Lemma 3.6.1.

□

This finishes the proof of the Lemma 5.2.4.

□

Now we are ready to prove the Theorem 5.2.1.

Proof of Theorem 5.2.1.

Recall that t denotes the smallest integer such that $f_H^*(K_t) = 1$. We divide the proof into two cases:

Case 1: $t > n/2$

In this case, we reduce the f_H to f_{S_p} for some $p = \Omega(n)$. Let ν_H denote the minimum vertex cover size of H. Since $t > n/2$, we have $\nu_H \leq n/2$. When $\nu_H = 1$ the property is trivially a star property and from the Lemma 5.2.2 we already get $Q(f_H) = \Omega(n)$. Otherwise we restrict f_H by picking a clique on $\nu_H - 1$ vertices and joining all the other $n - \nu_H + 1$ remaining vertices to each

[3] Since $x_{ij} = z_i.z_j$, every monomial of f_H^* of size d becomes a monomial of size at most $2d$ in f'.

vertex in this clique. The resulting function takes a graph on $p = n - \nu_H + 1$ vertices as input. Let's denote these vertices by S.

As the clique on $\nu_H - 1$ vertices cannot accommodate all the vertices in the minimum vertex cover of H, in order to satisfy the property f_H at least one vertex v in the vertex cover must occur among S. This vertex v may have some edges incident on the vertices of the clique and some edges incident on the vertices of S. In the restriction all the possible edges to the clique are already present. Thus as soon as we have the remaining edges to the vertices of S the property f_H is satisfied.

Hence the property is now reduced to finding a star graph with d edges, f_{S_d} where d is defined as follows: Let C be a vertex cover. Furthermore let $d_{out}(v)$ denote the number of neighbors of a vertex v in C that are outside C and $d_{out}(C)$ be the minimum over all such vertices v in C. Then d is the minimum $d_{out}(C)$ of a minimum vertex cover C of H (minimized over all the minimum vertex covers). Thus as soon as we have the star graph with d edges, our original restricted f_H is satisfied.

Now from the Lemma 5.2.2 we get $Q(f_H) = \Omega(n)$.

Case 2: $t \leq n/2$

Note that $t > \alpha_H$. And since $t \leq n/2$, we have $n - t = \Omega(n)$. Hence from the Lemma 5.2.4 we get the bound of $\Omega(\sqrt{t(n-t)})$, which is $\Omega(\sqrt{\alpha_H \cdot n})$.

□

As a result, we immediately get the following corollaries.

Corollary 5.2.6. *For any non-empty H,*

1. $Q(f_H) = \Omega\left(\frac{n}{\sqrt{d_{avg}(H)}}\right)$,
2. $Q(f_H) = \Omega\left(\frac{n}{\sqrt{\chi_H}}\right)$,
3. $Q(f_H) = \Omega\left(\sqrt{\frac{n}{p}}\right)$,

where $d_{avg}(H)$ denotes the average degree of the vertices of H, χ_H denotes the chromatic number of H, and p denotes the critical probability [FKW02a] of H.

Proof. (1) From Turán's theorem, we have: $\alpha_H \geq n/(2 \cdot d_{avg}(H))$.

(2) Since $\alpha_H \cdot \chi_H \geq n$ we have $\alpha_H \geq n/\chi_H$.

(3) Since the critical probability of H is p, the average degree of H is at most pn. Hence from Corollary 5.2.6(1), we get the $\Omega(n/p)$ bound. □

In particular, we get an $\Omega(n)$ bound when the graph H is sparse ($|E(H)| = O(n)$), or H has a constant chromatic number, or the critical probability of H is $O(1/n)$. Friedgut, Kahn, and Wigderson [FKW02a] show an $\Omega(n/p)$ bound for the randomized query complexity of general monotone properties. Quantization of this bound remains open. General monotone properties can be thought of as the Subgraph Isomorphism for a *family* of minimal subgraphs. The item 3 above, gives a quantization of [FKW02a] in the case when the family contains only a single subgraph.

We also get the following corollary.

Corollary 5.2.7. *For any non-empty H,*

$$Q(f_H) = \Omega(n^{3/4}).$$

.

Proof. When $d_{avg}(H) \geq \sqrt{n}$ the Lemma 3.3.4 gives an $\Omega(n^{3/4})$ bound. Otherwise when $d_{avg}(H) < \sqrt{n}$ we use the Corollary 5.2.6(1), which gives the same bound. □

Prior to this work only an $\Omega(n^{2/3})$ bound was known from the work of Santha and Yao [SY] on general monotone graph properties.

5.2.3 Subgraph Isomorphism for 3-Uniform Hypergraphs

In this section we extend the $\Omega(n^{3/4})$ bound for the Subgraph Isomorphism for graphs to the 3-uniform hypergraphs. In particular, we obtain an $\Omega(n^{4/5})$ bound for the Subgraph Isomorphism for 3-uniform hypergraphs, improving upon the $\Omega(n^{3/4})$ bound obtained via the minimal certificate size.

Theorem 5.2.8. *Let H be a non-empty 3-uniform hypergraph on n vertices. Then,*

$$Q(f_H) = \Omega(n^{4/5}).$$

Before going to the proof of Theorem 5.2.8, we extend the Lemma 5.2.4 to the 3-uniform hypergraphs. Let t be the smallest such that $f^{\star}_H(K_t) = 1$. Note that $t = \alpha_H + 1$.

Lemma 5.2.9. *Let H be a 3-uniform hypergraph on n vertices. Then:*

$$Q(f_H) \geq \Omega(\sqrt{t(n-t)}).$$

Proof. Let $T_n^t(z_1, \ldots, z_n)$ denote the threshold function on n variables that outputs 1 if and only if at least t variables are 1. We *embed* a T_n^t in f_H^* (on inputs of Hamming weight $t-1$ and t) via the following mapping: Let $x_{ijk} := z_i \cdot z_j \cdot z_k$. Let $f'(z_1, \ldots, z_n) := f_H^*(\{x_{ijk}\})$. Note that $f' \equiv T_n^t$. Also note[4] that the $\widetilde{deg}(f') \leq 3 \cdot \widetilde{deg}(f_H^*)$. Since $Q(f) \geq \widetilde{deg}(f)$, it remains to prove that $\widetilde{deg}(f') = \Omega(\sqrt{t(n-t)})$, which follows from the Lemma 3.6.1.

□

Now we give a proof of the Theorem 5.2.8.

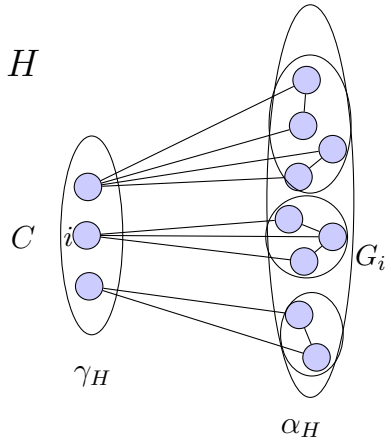


Figure 5.1: Structure of H

Proof of Theorem 5.2.8.

We divide the proof into two main cases.

Case 1: $\alpha_H > n/2$.

Let H be a 3-uniform hypergraph on n vertices. Let C denote a minimal vertex cover of H. Let $|C| = \nu_H$. Note that the hypergraph induced on $V - C$ is empty. For a vertex $i \in C$ let G_i denote the projection graph of the neighbors of i on $V - C$, i.e., $(i, u, v) \in E(H)$ (see Figure 5.1).

Let $\mathcal{P}_H$ denote the restriction of the f_H defined as follows: set the hyper-clique on $\nu_H - 1$ vertices to be present and add all the hyper-edges incident on the vertices of this clique. Let S denote the set of remaining $n - \nu_H + 1$ vertices. The hyper-edges among S are still undetermined. Note that $\mathcal{P}_H$ is a non-trivial property of $n-\nu_H+1$ vertex hypergraphs, since H cannot be contained in the $\nu_H - 1$ hyper-clique and edges incident on it as the minimum vertex cover size of H is ν_H.

[4]Since $x_{ijk} = z_i \cdot z_j \cdot z_k$, every monomial of f_H^* of size d becomes a monomial of size at most $3d$ in f'.

Lemma 5.2.10. *If* $\exists C, \exists i : |E(G_i)| = O(n^{7/5})$*, then* $Q(f_H) = \Omega(n^{4/5})$.

Proof. In this case $\mathcal{P}_H$ has a certificate of size $O(n^{7/5})$.

Now we use the following lemma due to Sun et al. [SYZ04b].

Lemma 5.2.11 (Transitive Packing [SYZ04b])**.** *Let* f *be a monotone transitive function on* n *variables. If* f *has a minimal certificate of size* s *then every certificate of* f^* *must have size at least* n/s.

Hence from the above Lemma 5.2.11 the certificate size of $\mathcal{P}^*_H$ is $\Omega(\frac{n^3}{n^{7/5}}) = \Omega(n^{8/5})$. Now from the Lemma 3.3.4 we get $Q(f_H) = \Omega(n^{4/5})$.

□

Hence from now onwards we assume that for all $i, |E(G_i)| = \Omega(n^{7/5})$. Moreover, we may also note that $\nu_H = O(n^{1/5})$, if not we have a minimal certificate for $\mathcal{P}_H$ of size $\Omega(n^{8/5})$. And hence from the Lemma 3.3.4 we already get the desired bound of $Q(f_H) = \Omega(n^{4/5})$.

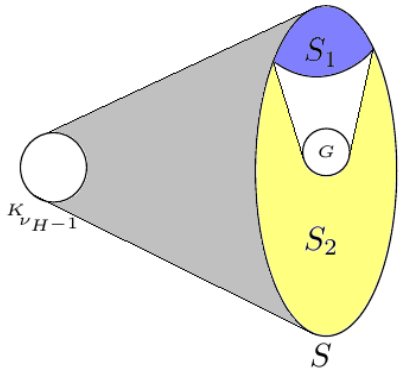


Figure 5.2: The Restriction $\mathcal{P}''$

Now we obtain a restriction $\mathcal{P}'$ of $\mathcal{P}_H$ as follows: divide S into two parts say S_1 and S_2 of size n_1 and n_2 respectively, where we choose $n_1 = \Theta(n^{1/5})$ and $n_2 = \Theta(n)$. Set all the hyper-edges within S_1 to be present and set all the hyper-edges within S_2 to be absent. Also set all the hyper-edges with two endpoints in S_1 and one in S_2 to be absent. Only possible undetermined hyper-edges are with one endpoint in S_1 and two in S_2. Note that even after setting all hyper edges in S_1 to be present we can safely assume that the property remains non-trivial. Otherwise we would have a certificate for $\mathcal{P}_H$ of size $O(n^{3/5})$, hence the dual will have large ($\gg \Omega(n^{8/5})$) certificates.

Let G be a projection graph among all the G_i's containing the least number of edges inside S_2. We further obtain a restriction $\mathcal{P}''$ by fixing a copy of G inside S_2 and allowing only potential

hyper-edges with one endpoint in S_1 and the other two endpoints forming an edge of G (see Figure 5.2).

Let C be a vertex cover of H of minimum cardinality. Note that in order to satisfy $\mathcal{P}_H$, at least one of the vertices from C must move to S. Let us call a vertex of C that moves to S as pivot. Let k be the largest integer such that P_H has a minimal certificate with k pivots. Note that from Lemma 5.2.10 each pivot has $\Omega(n^{7/5})$ edges incident on it. Therefore if $k > n_1/2$ then we already have a minimal certificate whose size is $\Omega(n^{8/5})$. Otherwise: $k \leq n_1/2$. First we argue that any pivot must belong to S_1. If on the contrary, it were in S_2 then the only possible edges incident on such a pivot v are of the form (v, u, w) where $u \in S_1$ and $w \in S_2$. But there can be at most $O(n^{6/5})$ such edges, which contradicts the fact that any pivot supports at least $\Omega(n^{7/5})$ edges. Let the degree of a pivot be the number of edges inside S_2 that are adjacent to it. Next we choose a certificate for $\mathcal{P}_H$ with at most $k \leq n_1/2$ pivots such that the degree of the minimum degree pivot is minimum possible. Then we leave aside the minimum degree pivot in this certificate and fix the $k-1$ other pivots and their projection on S_2. From each of the remaining $n_1 - k + 1$ vertices we keep the projection of the minimum degree pivot on S_2 as the only possible edges.

Now from minimality of our choice at least one of these vertices must have all these $\Omega(n^{7/5})$ edges in order for the original graph to contain H. Thus we get an $\bigvee_{\Omega(n^{1/5})} \bigwedge_{\Omega(n^{7/5})}$ function as the restriction.

Since an $OR \circ AND$ on m variables admits an $\Omega(\sqrt{m})$ lower bound on the quantum query complexity we get $Q(f_H) = \Omega(n^{4/5})$.

Case 2: $\alpha_H \leq n/2$.

In this case we use Lemma 5.2.9. Since $n - \alpha_H \geq n/2$, we immediately get $Q(f_H) = \Omega(\sqrt{\alpha_H \cdot n})$.

Now in order to prove Theorem 5.2.8, we need to show that the above bound always yields an $\Omega(n^{4/5})$ bound. Thus we further consider two cases based on the average degree. And in fact this gives us a larger $\Omega(n^{5/6})$ bound for the case 2.

Let d denote the average degree of H.

Case 2a: $d > n^{2/3}$.

In this case $|E(H)| > \Omega(n^{5/3})$. Hence from Lemma 3.3.4 we get an $\Omega(n^{5/6})$ bound.

Case 2b: $d \leq n^{2/3}$.

Before going into the proof of this case we state the following lemma due to Ajtai et al. [AKP+82].

Lemma 5.2.12 (Turán and Extended Turán [AKP+82]). *If the average degree of a graph G is d then G contains an independent set of size at least $\Omega(n/d)$.*

And if the average degree of a k-uniform hypergraph G is d then G contains an independent set of size at least $\Omega(n/d^{\frac{1}{k-1}})$.

Here we use the above extension of Turán's Theorem (see Lemma 5.2.12) to 3-uniform hypergraphs. Since the average degree is $O(n^{2/3})$, we get $\alpha_H \geq \Omega(n^{2/3})$. Therefore from Lemma 5.2.9 we get $Q(f_H) = \Omega(n^{5/6})$.

This completes the proof of Theorem 5.2.8.

□

In the following two sections we study the Subgraph Homomorphism Problem. We first prove the quantum query complexity lower bounds for graphs and then for 3-uniform hypergraphs.

5.3 Subgraph Homomorphism Problem

We also investigate a closely related Subgraph Homomorphism Problem for graphs and 3-uniform hypergraphs.

5.3.1 Subgraph Homomorphism Problem for Graphs

Recall that a homomorphism from a graph H into a graph G is a function $h : V(H) \to V(G)$ such that: if $(u, v) \in E(H)$ then $(h(u), h(v)) \in E(G)$.

Let $f_{[H]}$ be the function defined as follows: $f_{[H]}(G) = 1$ if and only if H admits a homomorphism into G.

Note that unlike the isomorphism, the homomorphism need not be an injective function from $V(H)$ to $V(G)$.

Here we show the following Theorem for the graphs:

Theorem 5.3.1. *For any non-empty H,*

$$Q(f_{[H]}) = \Omega(n).$$

Proof. Let $\chi(H)$ denote the chromatic number of H. Note that H has a homomorphism into K_t for $t = \chi(H)$, i.e., $f_{[H]}(K_{t-1}) = 0$ and $f_{[H]}(K_t) = 1$.

We consider the following two cases.

Case 1: $t \geq n/2$:

As K_{t-1} is a no instance and K_t is an yes instance for the property $f_{[H]}$, the minimum certificate size, $m(f_{[H]}) = \Omega(t^2) = \Omega(n^2)$. Hence from Lemma 3.3.4 we get an $\Omega(n)$ lower bound on the quantum query complexity.

Case 2: $t < n/2$:

Consider the following restriction: We set a clique K_{t-2} on $t-2$ vertices to be present and we also set all the edges from the remaining $n-t+2$ vertices to this clique to be present. Now notice that as soon as there is an edge between any two of the remaining $n-t+2$ vertices, we have a K_t. Hence the property $f_{[H]}$ has become the property of containing an edge among the $n-t+2$ vertices. Since $t < n/2$, this is an OR function on $\Omega(n^2)$ variables. Thus $Q(f_{[H]}) = \Omega(n)$.

□

Remark 5.3.2. *Our proof in fact shows that the minimum certificate size of either $f_{[H]}$ or $f^*_{[H]}$ is $\Omega(n^2)$. Hence we also obtain*

$$R(f_{[H]}) = \Omega(n^2).$$

.

5.3.2 Subgraph Homomorphism for 3-Uniform Hypergraphs

We now proceed to prove the quantum query complexity lower bound of the Subgraph Homomorphism Problem for 3-uniform hypergraph.

Theorem 5.3.3. *For any non-empty 3-uniform hypergraph H on n vertices:*

$$Q(f_{[H]}) = \Omega(n^{3/2}).$$

Proof. Proof of this theorem is similar to the proof of the previous theorem, Theorem 5.3.1.

Let $\chi(H)$ denote the chromatic number of H. A valid χ coloring of a hypergraph is a coloring function $f : V \to [\chi]$ such that no edge of the graph is monochromatic (i.e., all incident vertices of any edge are of same color). Then the chromatic number $\chi(H)$ of the hypergraph H is the minimum number χ such that a function $f : V \to [\chi]$ exists.

Note that H has a homomorphism into K_t for $t = \chi(H)$, i.e., $f_{[H]}(K_{t-1}) = 0$ and $f_{[H]}(K_t) = 1$.

We consider the following two cases.

Case 1: $t \geq n/2$:

Unlike the graph homomorphism case, we cannot claim the presence of a K_t in this case. However we can still use the following fact:

Fact 5.3.4. *(Alon [Alo85]) If H is a 3-uniform hypergraph which is not k colorable then*

$$|E(H)| = \Omega(k^3).$$

Therefore, the minimum certificate size $m(f_{[H]}) = \Omega(t^3) = \Omega(n^3)$. Hence from Lemma 3.3.4 we get an $\Omega(n^{3/2})$ lower bound on the quantum query complexity.

Case 2: $t < n/2$:

Consider the following restriction: We set a clique K_{t-3} on $t-3$ vertices to be present and we also set all the edges from remaining $(n-t+3)$ vertices to this clique to be present. Now notice that as soon as there is an edge between any three of the remaining $(n-t+3)$ vertices, we have a K_t. Hence the property $f_{[H]}$ has become the property of containing an edge among the $n-t+3$ vertices. Since $t < n/2$, this is an OR function on $\Omega(n^3)$ variables. Thus $Q(f_{[H]}) = \Omega(n^{3/2})$.

□

Chapter 6

Graph Properties in the Node Query Settings

This chapter is based on the following paper:

- *Balaji, N., Datta, S., Kulkarni, R., and Podder, S. (2015). Graph properties in node-query setting: effect of breaking symmetry. In Proceedings of the 41st International Symposium on the Mathematical Foundations of Computer Science (MFCS) 2016. [BDKP15]*

6.1 Introduction

In the Subgraph Isomorphism Problem (as discussed before) a graph H is known, and access to another graph G is given only via queries to the edges of G. The goal is to figure out whether G contains H as its subgraph. Note that it does not have to be induced. Now it is reasonable to ask what happens if we change the type of query and instead of asking whether an edge $e_i, i \in \binom{n}{2}$ is present in G we ask whether a *vertex* $v_j, j \in [n]$ is present in G. Thus any vertex can either be *active* or *inactive*. Naturally the set of all active vertices in G forms an induced subgraph. We can then ask if the induced subgraph contains the graph H as a subgraph or whether it has some property $\mathcal{P}$. Formally it is defined as follows: A graph $G = (V, E)$ and a property $\mathcal{P}$ are *fixed*. We have access to $S \subseteq V$ via queries of the form "Does i belong to S?". We are interested in the minimum number of queries needed in the worst case in order to determine whether $G[S]$ – the subgraph of G induced on S – satisfies property $\mathcal{P}$, which we denote by $cost(\mathcal{P}, G)$. Similar to the edge query setting this new setting is combinatorially very rich and it opens a new platform for asking interesting questions. We investigate this new setting in this chapter.

6.1.1 Graph Properties in Node-Query Setting

Graph properties in the edge query settings has been vastly investigated. In this chapter , we investigate the node query settings. It is defined as follows: A graph $G = (V, E)$ and a property $\mathcal{P}$ are fixed. We have access to $S \subseteq V$ via queries of the form "Does i belong to S?". We are interested in the minimum number of queries needed in the worst case in order to determine whether $G[S]$ – the subgraph of G induced on S – satisfies $\mathcal{P}$, which we denote by $cost(\mathcal{P}, G)$. One may define a similar notion of cost for randomized and quantum models.

We call G the base graph for $\mathcal{P}$. We say that a vertex i of G is relevant for $\mathcal{P}$ if there exists some S containing i such that exactly one of $G[S]$ and $G[S - \{i\}]$ satisfies $\mathcal{P}$. We say that G is relevant for $\mathcal{P}$ if all its vertices are relevant for $\mathcal{P}$. The minimum possible cost of $\mathcal{P}$, denoted by[1] $min\text{-}cost(\mathcal{P})$, is defined as follows:

$$min\text{-}cost_n(\mathcal{P}) = \min_G \{cost(\mathcal{P}, G) \mid G \text{ is relevant for } \mathcal{P} \ \& \ |V(G)| = n\}.$$

Note that in the node-query settings the notion of *relevance of a graph G for the property $\mathcal{P}$* is important because if any vertex $v \in G$ is *not* relevant then v cannot possibly influence the output of the function and hence any query algorithm do not need to query it.

Similarly, one can define $max\text{-}cost(\mathcal{P})$ as follows:

$$max\text{-}cost_n(\mathcal{P}) = \max_G \{cost(\mathcal{P}, G) \mid G \text{ is relevant for } \mathcal{P} \ \& \ |V(G)| = n\}.$$

The $max\text{-}cost$ is a more natural notion of complexity when one is interested in studying the universal upper bounds. Investigating the $max\text{-}cost$ in our setting is a topic of future research and we shall discuss this in the conclusion section (see Chapter 9). For the time being, the notion of $min\text{-}cost$ will be more relevant.

Notice that in the definition of $min\text{-}cost$ we take the minimum over all possible graphs that are relevant. We can define the notion of $\mathcal{G}\text{-}min\text{-}cost(\mathcal{P})$ where we restrict ourself to only a certain class of graphs $\mathcal{G}$. For example, we can consider the set of planar graphs or graphs with a certain amount of symmetry and ask what is the minimum cost of finding a property $\mathcal{P}$ in planar graphs or for graphs with symmetry, $\mathcal{G}\text{-}min\text{-}cost(\mathcal{P})$.

[1] We slightly abuse this notation by omitting the subscript n.

Formally,

$$\mathcal{G}\text{-}min\text{-}cost_n(\mathcal{P}) = \min_{G \in \mathcal{G}}\{cost(\mathcal{P}, G) \mid G \text{ is relevant for } \mathcal{P} \ \& \ |V(G)| = n\}.$$

Similar to the edge query model, there are several lines of investigation that can be done in the node query setting. We choose to focus on the effect of removing symmetry from graphs and study how low the query complexity can fall if we take away the symmetry assumption from the input graph. By taking away the symmetry we mean, first we ask what is the $\mathcal{G}\text{-}min_n\text{-}cost(\mathcal{P})$ when $\mathcal{G}$ is a set of graphs with a certain amount of symmetry. Then we compare it with the $\min\text{-}cost_n(\mathcal{P})$ where there is no restriction on the $\mathcal{G}$[2] [3].

6.1.2 Some Practical Applications of Node-Query Setting

It appears that the node-query setting is also a natural abstraction of scenarios where one is interested in the properties of subgraph induced by active nodes in a network. We discuss three such examples below.

1. Consider a graph that models the associations in a social network, say the Facebook graph (where two nodes are adjacent if they are Facebook friends). At any given time, users can be online or offline. We might be interested in finding out if there is any user who is online and is influential, in the sense that he/she has many neighbors (friends) who are also online at that time. This problem can be formulated in our setting as whether the induced subgraph has a vertex of large degree or not (Section 6.3.3).

2. Consider a physical network with several nodes and links between them. At any given time, the nodes of the network can be either active or inactive. One way to find out if a node is active is to ping it (possibly by physically going to the site), which comes with some fixed cost. For example, the underlying network could be the network of routers which are physically connected by wires. Some of the routers may go on and off over time. At any given time, we want to know whether a message can be sent between two specified nodes via the active routers. This problem can be formulated in our setting as whether the subgraph induced by active routers has a path between two specified nodes s and t or not. (Section 6.5.1)

[2] In our setting, every hereditary property is a monotone Boolean function.

[3] We would like to highlight that although we didn't explicitly define $min\text{-}cost(\mathcal{P})$ or $max\text{-}cost(\mathcal{P})$ for randomized query model, all our lower bound proofs are based on sensitivity arguments and hence work even for randomized case.

3. Consider a chemical lab which performs experiments with certain basic ingredients to build medicines. Suppose a concoction comes out of an experiment and one wants to know whether it is harmful or not. There are tests available for testing the presence of an ingredient in the concoction. The lab also has a table of which two ingredients together form a harmful combination. So the goal is to perform the tests for presence of individual ingredients to check if any of the harmful combination is present. This problem can be formulated in our setting as whether the induced subgraph is empty or not. (Section 6.3.3)

To the best of our knowledge, no systematic study of node-query setting has been yet undertaken. Here we initiate such a line of inquiry for graph properties. In particular, we focus on the role of presence and absence of *symmetry*.

Related Work

Understanding the effect of symmetry on computation is a very well-studied theme in the literature. In the context of query complexity, there has been a substantial amount of effort invested in understanding the role of symmetry on computation. A recurrent theme has been to exploit the symmetry and/or some other structure [KS13] of the underlying functions to prove good lower bounds on their query complexity.

One such line of research concerns with the famous Andera-Rosenberge-Karp Conjecture [KSS84] which asserts that every non-trivial monotone graph property of n vertex graphs (in the edge-query model) must be evasive, i.e., its query complexity is $\binom{n}{2}$. While a weaker bound of $\Omega(n^2)$ is known, the conjecture remains widely open to this date. Even the randomized query complexity of monotone graph properties is also conjectured to be $\Omega(n^2)$ [FKW02b] and we are far from proving it. Thus it make sense to try to prove the conjectures for a certain class of graphs, for example graphs with symmetry. Sun,Yao, and Zhang [SYZ04a] studied query complexity of graph properties and several transitive functions in the edge query setting. Their motivation was to investigate how low can the query complexity fall if we drop the assumption of monotonicity or lower the amount of symmetry. In this work, we follow their footsteps and ask what happen if we *totally* drop the symmetry assumption. In the past, Lovász has also conjectured [Iva] that checking independence of S exactly in the node query setting is evasive. The main difference between the past works and this one is that most of the previous work exploit the symmetry to prove (or to conjecture) a good lower bound, whereas we investigate the consequences of breaking the symmetry for the query complexity.

6.1.3 Effect of Breaking Symmetry

It turns out that in the node query setting, when $\mathcal{G}$ has the highest amount of symmetry, i.e., when $\mathcal{G}$ is the class of complete graphs, then for every hereditary property $\mathcal{P}$, $\mathcal{G}$-$min\text{-}cost(\mathcal{P})$ is nearly the worst possible, i.e., $\Omega(n)$. Recall that a hereditary property is a property of graphs, which is closed under deletion of vertices as well as edges. For instance acyclicity, bipartiteness, planarity, and triangle-freeness are hereditary properties whereas connectedness and containing a perfect matching are not. Every hereditary property can be described by a (not necessarily finite) collection of its forbidden subgraphs. In our setting, every hereditary property is a monotone Boolean function. It also turns out that one does not require the whole symmetry of the complete graph to guarantee the $\Omega(n)$ bound. Even for weaker symmetry assumptions on graphs i.e., when $\mathcal{G}$ is the class of transitive graphs, for any hereditary property $\mathcal{P}$ the $\mathcal{G}$-$min\text{-}cost(\mathcal{P})$ would remain the highest possible, i.e., n. So for the complexity to fall down substantially we let go of the transitivity of $\mathcal{G}$ and take $\mathcal{G}$ to be the class of all graphs: $\mathcal{G}$-$min\text{-}cost(\mathcal{P})$ = $min\text{-}cost(\mathcal{P})$.

So how low can $min\text{-}cost(\mathcal{P})$ fall in the absence of any symmetry? This is the main question addressed by our work. We show that for any hereditary property $\mathcal{P}$, the $min\text{-}cost(\mathcal{P})$ falls down at least quadratically, i.e., to $O(\sqrt{n})$. For some properties, it can fall even further below (polynomially down) with polynomials of arbitrary constant degree, In particular we show that for any hereditary graph property $\mathcal{P}$: $min\text{-}cost(\mathcal{P}) = O(n^{1/(d_{\mathcal{P}}+1)})$, where $d_{\mathcal{P}}$ is the minimum degree of the minimum forbidden subgraph of the property $\mathcal{P}$.

The main question left open by our work is: does there exist a hereditary property $\mathcal{P}$ for which $min\text{-}cost(\mathcal{P})$ is exponentially low? In other words:

Question 6.1.1. *Is it true that for every hereditary property $\mathcal{P}$ there exists an integer $k_{\mathcal{P}} > 0$ such that*

$$min\text{-}cost(\mathcal{P}) = \Omega(n^{1/k_{\mathcal{P}}})?$$

As a partial answer to this question we prove that, when H is a fixed graph on k vertices and $\mathcal{P}_H$ denotes the property of containing H as a subgraph, then, $min\text{-}cost(\mathcal{P}_H) = \Omega(n^{1/k})$. Furthermore a non-constant lower bound holds for *any* hereditary property i.e., for any hereditary graph property $\mathcal{P}$, $min\text{-}cost(\mathcal{P}) = \Omega\left(\frac{\log n}{\log\log n}\right)$.

Here in this chapter we study several other specific properties like Triangle-freeness, Independence etc. and proved tight bounds. We also consider restricted graph classes and prove several

new results.

Hereditary Graph Properties

Before going any further, we formally recall the definition of Hereditary graph properties.

Definition 6.1.2 (Hereditary graph properties)**.** *A property $\mathcal{P}$ of graphs is simply a collection of graphs. The members of $\mathcal{P}$ are said to satisfy $\mathcal{P}$ and non-members are said to fail $\mathcal{P}$. A property is hereditary if it is closed under deletion of vertices as well as edges. For instance: acyclicity, planarity, and 3-colorability are hereditary properties, whereas connectivity and containing a perfect matching are not. On the other hand, vertex-hereditary is closed only under vertex-deletion (e.g. being chordal). Every hereditary property $\mathcal{P}$ can be uniquely expressed as a (possibly infinite) family $\mathcal{F}_{\mathcal{P}}$ of its forbidden subgraphs. For instance: acyclicity can be described as forbidding all cycles. Given a graph G, the hitting set $S_{G,\mathcal{P}}$ for $\mathcal{P}$ is a subset of $V(G)$ such that removing $S_{G,\mathcal{P}}$ from G would make the property $\mathcal{P}$ present*[4]*. Hereditary graph properties in node-query setting are monotone decreasing Boolean functions. Sometimes we refer hereditary properties by their negation. For instance: containing triangle.*

6.2 Presence of Symmetry: Does it Guarantee Weak-Evasiveness?

Our goal is to study how low can query complexity fall in the absence of any symmetry assumption of the base graph. First we show how much does the symmetry assumption lift up the complexity.

In edge-query setting, Aanderaa-Rosenberg-Karp Conjecture [KSS84, CKS01] asserts that any non-trivial monotone graph property must be evasive, i.e., one must query all $\binom{n}{2}$ edges in worst-case. The following generalization of the ARK Conjecture asserts that only monotonicity and modest amount of symmetry, namely transitivity, suffices to guarantee the evasiveness [Lut01].

Conjecture 6.2.1 (Evasiveness Conjecture)**.** *Any non-constant monotone transitive function f on n variables has $D(f) = n$.*

This conjecture appears to be notoriously hard to prove even in several interesting special cases. Recently Kulkarni [Kul13] formulates:

Conjecture 6.2.2 (Weak Evasiveness Conjecture)**.** *If f_n is a sequence of monotone transitive*

[4] such that every graph in $\mathcal{F}_{\mathcal{P}}$ shares a node with $S_{\mathcal{G},\mathcal{P}}$.

functions on n variables then for every $\epsilon > 0$:

$$D(f_n) = \Omega(n^{1-\epsilon}).$$

Although Weak EC appears to be seemingly weaker, Kulkarni [Kul13] observes that it is equivalent to the EC itself. His results hint towards the possibility that disproving Weak EC might be as difficult as separating TC^0 from NC^1. However: proving special cases of Weak EC appears to be relatively less difficult. In fact, Rivest and Vuillemin [RV76] confirm the Weak EC for graph properties and recently Kulkarni, Qiao, and Sun [KQS15] confirm Weak EC for 3-uniform hyper graphs and Black [Bla15] extends this result to k-uniform hyper graphs. All these results are studied in the edge-query setting. It is natural to ask whether the Weak EC becomes tractable in node-query setting. The monotone functions in node-query setting translate precisely to the hereditary graph properties. Here we show that it does become tractable for several hereditary graph properties. But first we need the following lemma [Cha05, SYZ04a]:

Lemma 6.2.3. *Let f be a non-trivial monotone transitive function. Let k be the size of a 1-input with minimal number of 1s. Then: $D(f) = \Omega(n/k^2)$.*

Let $\mathcal{G}_\mathcal{T}$ denote the class of transitive graphs. Let H be a fixed graph. Let $\mathcal{P}_H$ denote the property of containing H as a subgraph. The following theorem directly follows from Lemma 6.2.3.

Theorem 6.2.4.

$$\mathcal{G}_\mathcal{T}\text{-}min\text{-}cost(\mathcal{P}_H) = \Omega(n).$$

The above result can be generalized for any finite family of forbidden subgraphs. We do not yet know how to prove it for infinite family in general. However below we illustrate a proof for one specific case when the infinite family is the family of cycles. First we need the following lemma:

Lemma 6.2.5. *Let G be a graph on n vertices, m edges, and maximum degree d_{max}. Let $\mathcal{C}$ denote the property of being acyclic. Then,*

$$cost(\mathcal{C}, G) \geq (m-n)/d_{max}.$$

Proof. To make G acyclic one must remove at least $m-n$ edges. Removing one vertex can remove at most d_{max} edges. Thus the size of minimum feedback vertex set (FVS) is at least $(m-n)/d_{\max}$.

The adversary answers all vertices outside this FVS to be present. Now the algorithm must query every vertex in the minimum FVS. □

Theorem 6.2.6.

$$\mathcal{G}_{\mathcal{T}}\text{-}min\text{-}cost(\mathcal{C}) = \Omega(n).$$

Proof. Since G is transitive, G is d regular for some d [GR13]. Therefore $m = dn/2$ and $d_{max} = d$. Hence from Lemma 6.2.5 we get the desired bound for the case when $d > 2$.

When $d = 2$ the base graph is a 2-regular graph and any 2-regular graph consists of one or more (disconnected) cycles. Thus for each vertex in every cycle adversary would keep saying 1 until the last vertex in that cycle. Thus we need to query all the vertices in the graph. □

We also show similar bound for the property of being planar:

Lemma 6.2.7. *Let G be a graph on n vertices, m edges, and maximum degree d_{max}. Let $\mathcal{P}'$ denote the property of being planar. Then,*

$$cost(\mathcal{P}', G) \geq (m - 3n + 6)/d_{max}.$$

Proof. To make G planar one has to remove at least $(m - 3n + 6)$ edges from the graph G. Removing one vertex can remove at most d_{max} edges. Thus the size of minimum hitting set of G is at least $(m - 3n + 6)/d_{\max}$. The adversary answers all vertices outside this minimum hitting set to be present. Now the algorithm must query every vertex in the minimum hitting set. □

Theorem 6.2.8.

$$\mathcal{G}_{\mathcal{T}}\text{-}min\text{-}cost(\mathcal{P}') = \Omega(n).$$

Proof. Since G is transitive, G is d regular for some d [GR13]. Therefore $m = dn/2$ and $d_{max} = d$. Hence for $d \geq 7$ using Lemma 6.2.7 we get the desired bound. Note that as for this particular proof to work, we require $d \geq 7$, this theorem only holds for a class of transitive graphs with degree more than or equal to 7. □

Following special case of Weak EC remains open:

Conjecture 6.2.9. *For any hereditary property $\mathcal{P}$, for any $\epsilon > 0$:*

$$\mathcal{G}_{\mathcal{T}}\text{-}min\text{-}cost(\mathcal{P}) = \Omega(n^{1-\epsilon}).$$

6.3 Absence of Symmetry: How Low Can Query Complexity Fall?

In the previous section we saw that in the presence of a minimum symmetry assumption on the base graph the query complexity of any hereditary graph property is worst possible, i.e., n. In this section we shall investigate how much can the query complexity fall in the absence of any symmetry assumption on the base graph.

6.3.1 A General Upper Bound

Let $\mathcal{P}$ be a hereditary graph property and $d_{\mathcal{P}}$ denote the minimum possible degree of a minimal forbidden subgraph for $\mathcal{P}$.

Theorem 6.3.1. *For any hereditary graph property $\mathcal{P}$:*

$$min\text{-}cost(\mathcal{P}) = O(n^{1/(d_{\mathcal{P}}+1)}).$$

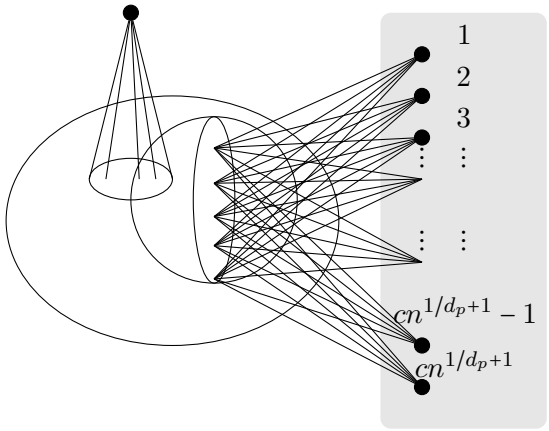


Figure 6.1: Construction of G for a General Upper Bound

Proof. Let $k = c \cdot n^{1/(d_{\mathcal{P}}+1)}$ where we choose the constant c appropriately. Construct a graph G on n vertices as follows (see Figure 6.1):

- Start with a clique on vertices $v_1, \ldots, v_k$.
- For every $S \subseteq [k]$ such that $|S| = d_{\mathcal{P}}$
  - add k new vertices $u_1^S, \ldots, u_k^S$ and
  - connect each u_i^S to every $v_i : i \in S$.

Note that every vertex v of G is relevant for $\mathcal{P}$. Otherwise the algorithm immediately knows that v cannot possible be a part of the property $\mathcal{P}$ and hence skips querying that vertex v. Now we describe an algorithm (see Algorithm 1) to determine $\mathcal{P}$ in $O(n^{1/(d_{\mathcal{P}}+1)})$ queries. Let $c_{\mathcal{P}}$ denote the smallest integer such that the clique on $c_{\mathcal{P}}$ vertices satisfies $\mathcal{P}$.

Algorithm 1:

- Query $v_1, \ldots, v_k$.
- If at least $c_{\mathcal{P}}$ of these vertices are present then $\mathcal{P}$ must fail.
- Otherwise there are at most $c_{\mathcal{P}} - 1$ vertices present (wlog: $v_1, \ldots, v_{c_{\mathcal{P}}-1}$).
  - For every subset $S \subseteq [c_{\mathcal{P}} - 1]$ such that $|S| = d_{\mathcal{P}}$, query $u_1^S, \ldots, u_k^S$.
  - If the graph induced on the nodes present (after all these $\binom{c_{\mathcal{P}}-1}{d_{\mathcal{P}}} \times k$ queries) satisfies $\mathcal{P}$ then answer Yes.
    Otherwise answer No.

Note that any vertex that is not queried by the above algorithm can have at most $d_{\mathcal{P}} - 1$ edges to the vertices in the clique $v_1, \ldots, v_k$. Since $d_{\mathcal{P}}$ is the minimum degree of a minimal forbidden subgraph for $\mathcal{P}$, these vertices now become irrelevant for $\mathcal{P}$. Thus the algorithm can correctly declare the answer based on only the queries it has made. It is easy to check that the query complexity of the above algorithm is $O(k)$ which is $O(n^{1/(d_{\mathcal{P}}+1)})$. □

Observe that $d_{\mathcal{P}} \geq 2$ for any non-trivial $\mathcal{P}$. Thus we immediately have the following corollary.

Corollary 6.3.2. *For any hereditary graph property $\mathcal{P}$:*

$$\textit{min-cost}(\mathcal{P}) = O(\sqrt{n}).$$

Theorem 6.3.1 and Corollary 6.3.2 show that in the absence of any symmetry on the graph G the query complexity can fall as low as $O(n^{1/(d+1)})$ where d denotes the minimum possible degree of a minimal forbidden sub-graph for $\mathcal{P}$. In particular, every hereditary property benefits at least quadratically.

We note that the above upper bound does not hold for general graph properties. For instance Connectivity has cost $\Theta(n)$, so does containment of a Perfect Matching, which are both non-hereditary properties (see Section 6.5.1).

6.3.2 General Lower Bounds

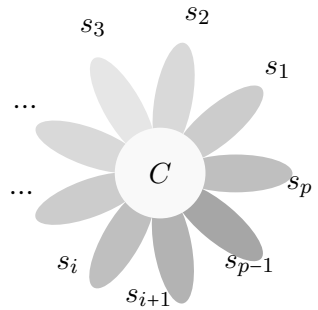
As a partial answer to the Question 6.1.1 we prove the following theorem.

Theorem 6.3.3. *Let H be a fixed graph on k vertices and let $\mathcal{P}_H$ denote the property of containing H as a subgraph. Then,*

$$min\text{-}cost(\mathcal{P}_H) = \Omega(n^{1/k}).$$

Moreover it generalizes to any fixed number of forbidden subgraphs each on at most k vertices. This implies that any hereditary property with *finitely many forbidden subgraphs* has a lower bound of $\Omega(n^{1/k})$, for a constant k depending only on the property.

To prove Theorem 6.3.3 we need the following definition of a sunflower.



Definition 6.3.4 (Sunflower)**.** *A sunflower with core set C and p petals is a collection of sets $S_1, \ldots, S_p$ such that for all $i \neq j$: $S_i \cap S_j = C$. Moreover, we want the petals to be non-empty.*

We use the following lemma due to Erdös and Rado [ER60].

Lemma 6.3.5 (Sunflower Lemma)**.** *Let $\mathcal{F}$ be a family of sets of cardinality k each. If $|\mathcal{F}| > k!(p-1)^k$ then $\mathcal{F}$ contains a sunflower with p petals.*

Proof of Theorem 6.3.3: Let G be a graph on n vertices such that every vertex of G is relevant for the property of containing H. Let

$$\mathcal{F} := \{S \mid |S| = k \ \& \ H \text{ is a subgraph of } G[S]\}.$$

Since every vertex of G is relevant for $\mathcal{P}_H$, we have: $|\mathcal{F}| \geq n/k$. Now from Lemma 6.3.5 we can conclude that $\mathcal{F}$ contains a sunflower on at least $|\mathcal{F}|^{1/k}/k = \Omega(n^{1/k})$ petals. Let C be the core of this sunflower. We consider the restriction of $\mathcal{P}_H$ on G where every vertex in C is present. Since $|C| < k$, $G[C]$ does not contain H. Now it is easy to check that one must query at least one vertex from each petal in order to determine $\mathcal{P}_H$.

□

We note that both Theorem 6.3.1 and Theorem 6.3.3 are not tight. However, we do prove tight bounds for several hereditary properties. We summarize few such interesting bounds in the Table below.

	Properties		With Symmetry	Without Symmetry
Finite	Independence/Emptiness	[Thm. 6.3.7]	$\Theta(n)$	$\Theta(\sqrt{n})$
	Bounded Degree	[Thm. 6.3.10]	$\Theta(n)$	$\Theta(\sqrt{n})$
	Triangle-freeness	[Thm. 6.3.8]	$\Theta(n)$	$\Theta(n^{1/3})$
	Containing K_t	[Thm. 6.3.1][Thm. 6.3.3]	$\Theta(n)$	$\Theta(n^{1/t})$
	Containing P_t	[Thm. 6.3.1][Thm. 6.3.9]	$\Theta(n)$	$\Theta(\sqrt{n})$
	Containing C_t	[Thm. 6.3.1][Thm. 6.3.9]	$\Theta(n)$	$\Theta(n^{1/3})$
	Containing H: $V(H) = k$	[Thm. 6.2.4][Thm. 6.3.1][Thm. 6.3.3]	$\Theta(n)$	$O(n^{1/(d_{min}+1)}), \Omega(n^{1/k})$
Infinite	Acyclicity	[Thm. 6.2.6][Thm. 6.3.1]	$\Theta(n)$	$O(n^{1/3})$
	Bi-partiteness	[Thm. 6.3.1]	Open	$O(n^{1/3})$
	3-colorability	[Thm. 6.3.1]	Open	$O(n^{1/4})$
	Planarity	[Thm. 6.2.8][Thm. 6.3.1]	$\Theta(n)$[5]	$O(n^{1/4})$

Table 6.1: Summary of Results for Finite/Infinite Forbidden Subgraphs

Using similar technique we get a non-constant lower bound, which holds for *any* hereditary property. Our proof again relies on the Sunflower Lemma.

Theorem 6.3.6. *For any hereditary graph property $\mathcal{P}$*

$$min\text{-}cost(\mathcal{P}) = \Omega\left(\frac{\log n}{\log\log n}\right).$$

Proof. Let G be a graph on n vertices such that every vertex of G is relevant for $\mathcal{P}$. Let k be the largest integer such that G contains a minimal forbidden subgraph for $\mathcal{P}$ on k vertices. Note that we consider vertex minimal forbidden subgraph.

Case 1: $k \geq \frac{\log n}{2\log\log n}$.

Since one must query all the vertices of a minimal forbidden subgraph, we obtain a lower bound of $k = \Omega(\log n/\log\log n)$.

Case 2: $k < \frac{\log n}{2\log\log n}$.

[5] when $d(G) \geq 7$.

Since every vertex of G is relevant for $\mathcal{P}$ and all the minimal forbidden subgraphs of $\mathcal{P}$ present in G are of size at most k, every vertex of G must belong to some minimal forbidden subgraph of size at most k. Consider the property $\mathcal{P}_k$ obtained from $\mathcal{P}$ by omitting the minimal forbidden subgraphs of $\mathcal{P}$ on k or more vertices. Our simple but crucial observation is that $\mathcal{P}$ and $\mathcal{P}_k$ are equivalent as far as G is concerned. Therefore, they have the same complexity. Now we define $\mathcal{F}_i$ for $i \leq k$ as follows:

$$\mathcal{F}_i := \{S \mid |S| = i \ \& \ G[S] \notin \mathcal{P} \ \& \ \forall T \subset S : G[T] \in \mathcal{P}\}.$$

Since every vertex of G is relevant for $\mathcal{P} \equiv \mathcal{P}_k$, we have: $|\bigcup_{i=1}^{k} \mathcal{F}_i| \geq n/k$. Since $\mathcal{F}_i$ and $\mathcal{F}_j$ are disjoint when $i \neq j$, we have $\sum_{i=1}^{k} |\mathcal{F}_i| \geq n/k$. Therefore one of the $\mathcal{F}_i$s must be of size at least n/k^2. We denote that $\mathcal{F}_i$ by $\mathcal{F}'$.

Now from Lemma 6.3.5 we can conclude that $\mathcal{F}'$ contains a sunflower on at least $|\mathcal{F}'|^{1/k}/k$ petals. Let C be the core of this sunflower. We consider the restriction of $\mathcal{P}$ on G where every vertex in C is present. Since $|C| < i$, by definition of $\mathcal{F}_i$ we must have $G[C] \in \mathcal{P}$. Now it is easy to check that one must query at least one vertex from each petal in order to determine $\mathcal{P}$. A simple calculation yields that one can obtain a lower bound of $\min\{k, \frac{2^{\Omega(\log n/k)}}{k}\}$. When $k = \log n/(2 \log\log n)$, this gives us $\Omega(\log n/\log\log n)$ bound.

□

6.3.3 Some Tight Bounds

We manage to show that Theorem 6.3.1 is tight for several special properties like Independence, Triangle-freeness, Bounded-degree etc. In order to prove the tight bounds, we show several inequalities which might be of independent interest combinatorially. We present one such inequality in Theorem 6.3.13. But before that we prove tight lower bounds for the above mentioned properties.

Independence/Emptiness

Theorem 6.3.7. *Let $\mathcal{I}$ denote the property of being an independent subset of nodes (equivalently the property of being an empty graph). Then,*

$$\textit{min-cost}(\mathcal{I}) = \Omega(\sqrt{n}).$$

Proof. Let G be a graph without isolated vertices. If $\chi(G) \leq \sqrt{n}$ then from Theorem 6.3.13 we get $\Omega(\sqrt{n})$ lower bound. Otherwise G must have a vertex of degree $\Omega(\sqrt{n})$. In this case the adversary answers this vertex to be present. Now the algorithm must query all its neighbours. □

Triangle-Freeness

Theorem 6.3.8. *Let $\mathcal{T}$ denote the property of being triangle-free. Then,*

$$min\text{-}cost(\mathcal{T}) = \Omega(n^{1/3}).$$

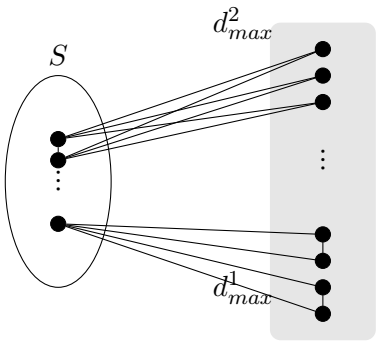


Figure 6.2: Tight Lower Bound for Triangle-Freeness

Proof. Let G be a graph such that every vertex belongs to some triangle. Let S denote a minimal hitting-set for all triangles, i.e., every triangle must share a vertex with S. Let d^1_{max} denote the maximum number of triangles supported at a vertex in G, i.e., maximum number of triangles whose common intersection is that vertex. Similarly, let d^2_{max} denote the maximum number of triangles supported at an edge in G. We consider the following cases:

Case 1: $|S| = \Omega(n^{1/3})$. As the set S has large size, the adversary answers all the vertices outside the hitting set to be present. Note that as no vertices of the hitting set are queried yet, any triangle is not yet present. Now as soon any of the vertex in S is present we have a triangle. This gives again $\Omega(n^{1/3})$ bound.

Case 2: $|S| = O(n^{1/3})$. As the size of the hitting set S is small there are $\Omega(n)$ vertices outside S. Moreover, as each vertex is relevant for the property each vertex outside S must be part of some triangle in G. Hence there must be an edge or a vertex inside S that forms triangles with $\Omega(n^{2/3})$ of the vertices outside S. Thus we have the following two subcases.

Case 2a: A vertex $v \in S$ forms triangles with $\Omega(n^{2/3})$ vertices outside S: Then $d^1_{max} \geq \Omega(n^{2/3})$. Thus the adversary can answer the vertex v to be present. Now consider the graph induced on the remaining endpoints of each the d^1_{max} triangles. Note that this graph does not contain any isolated vertices and number of vertices of this induced graph is also $\Omega(n^{2/3})$. Also note that this graph contains an edge if and only if we have a triangle in the original graph with the common vertex. Hence from Theorem 6.3.7 we get a bound of $\sqrt{d^1_{max}} = \Omega(n^{1/3})$.

Case 2b: $d^2_{max} \geq \Omega(n^{1/3})$: The adversary can answer both the endpoints of the common edge to be present, which would force the algorithm to query each of the remaining d^2_{max} vertices. This gives $\Omega(n^{1/3})$ bound.

□

Containing Path of Length t, P_t and Cycle of Size t, C_t

With a similar techniques it is easy to show the following theorem.

Theorem 6.3.9. *Let $\mathcal{P}_t$ denote the property of containing a path of length t, and let $\mathcal{C}_t$ denote the property of containing a cycle of size t. Then,*

$$\textit{min-cost}(\mathcal{P}_t) = \Omega(\sqrt{n}) \textit{ and}$$

$$\textit{min-cost}(\mathcal{C}_t) = \Omega(n^{1/3}).$$

Bounded Degree

Theorem 6.3.10. *Let $\mathcal{B}_d$ denote the property of having maximum degree at most d, where d is a constant. Then,*

$$\textit{min-cost}(\mathcal{B}_d) = \Omega(\sqrt{n}).$$

Proof. Let G be a graph such that every vertex belongs to some d-star (d vertices incident on a single vertex). Let S denote the hitting set for all stars of size d in G. Let d_{max} denote the maximum degree of G.

When $d_{max} \geq \sqrt{n}/10d$: The adversary answers the maximum degree vertex to be present. Now one must query $\Omega(d_{max})$ of its neighbors. And when $|S| \geq \sqrt{n}/10d$: The adversary answers all vertices outside the hitting set to be present. One must query the entire hitting set. Finally: we

claim that one of the above two cases must happen. Otherwise we have at most $n/100d^2$ neighbors of the hitting set, each of them can have at most $d-1$ other neighbours. This leaves some vertex t not in d-star.

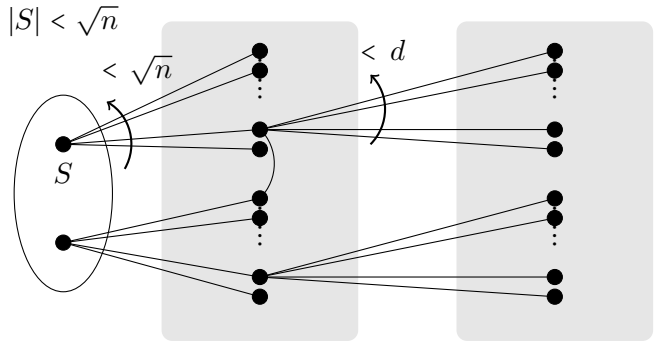


Figure 6.3: Tight Lower Bound for Bounded Degree

□

In fact the proof above generalizes to *local properties* of bounded degree graphs. For a property $\mathcal{P}$ we define $\mathcal{P}_L$ as follows:

Definition 6.3.11 (Local Property)**.** *G satisfies $\mathcal{P}_L$ if and only if for every vertex of G the graph induced on its neighbors satisfies $\mathcal{P}$.*

For instance: bipartite graphs are locally acyclic. It turns out that $\mathcal{P}_L$ is hereditary for any hereditary $\mathcal{P}$. Moreover, every graph in the forbidden family $\mathcal{F}_{\mathcal{P}_L}$ has a universal vertex, i.e., a vertex adjacent to all other vertices.

Theorem 6.3.12. *For any hereditary $\mathcal{P}$*

$$min\text{-}cost(\mathcal{P}_L \wedge \mathcal{B}_d) = \Omega(\sqrt{n}).$$

Proof. Let G be a graph such that every vertex belongs to some d-star (d vertices incident on a single vertex) or some $H \in \mathcal{F}_{\mathcal{P}_L}$. Let S denote the hitting set for $\mathcal{F}_{\mathcal{P}_L} \cup \{S_d\}$ in G. Let d_{max} denote the maximum degree of G.

When $d_{max} \geq \sqrt{n}/10d$, the adversary answers the maximum degree vertex to be present. Now one must query $\Omega(d_{max})$ of its neighbors (since there are d_{max}/d disjoint blocks). When $|S| \geq \sqrt{n}/10d$, the adversary answers all vertices outside the hitting set to be present. One must

query the entire hitting set. Finally: we claim that one of the above two cases must happen. Since every vertex belongs to some d-star or some $H \in \mathcal{F}_{\mathcal{P}_L}$ and every $H \in \mathcal{F}_{\mathcal{P}_L}$ has a universal vertex, we have that every vertex in G is reachable to some vertex in S by a path of length at most 2. Otherwise we have at most $n/100d^2$ neighbors of the hitting set, each of which can have at most $d-1$ other neighbours. This leaves some vertex t not in $\mathcal{F}_{\mathcal{P}_L} \cup \{S_d\}$. □

Lower Bound Based on the Chromatic Number

Theorem 6.3.13. *Let $\mathcal{I}$ denote the property of being an independent subset of nodes (equivalently the property of being an empty graph). Then,*

$$\mathcal{G}\text{-}min\text{-}cost(\mathcal{I}) \geq n/\chi$$

where χ is the chromatic number of a graph $G \in \mathcal{G}$.

Proof. Let $G \in \mathcal{G}$ be a graph on n vertices such that every vertex of G is relevant for $\mathcal{I}$, i.e., G does not contain any isolated vertices. Consider a coloring of vertices of G with χ colors. Let C_i denote the set of vertices colored with color i. We pick a coloring that maximizes $\max_{i \leq \chi}\{|C_i|\}$. Let C_{max} denote such a color class with maximum number of vertices in this coloring. Thus $|C_{\max}| \geq n/\chi$.

When $|C_{\max}| \leq (1 - \frac{1}{\chi})n$, the adversary answers all the vertices in $C_{\max}$ to be present. Since $C_{\max}$ is maximal and G does not contain any isolated vertices, every vertex outside $C_{\max}$ must be connected to some vertex in $C_{\max}$. As long as any of these outside vertices are present there will be an edge. Hence we get a lower bound of $n - |C_{\max}| \geq n/\chi$.

Now when $|C_{\max}| > (1 - \frac{1}{\chi})n$, since there are no isolated vertices in G, every vertex in $C_{\max}$ must have an edge to some vertex in $C_i \neq C_{\max}$. Furthermore as $|C_{\max}| > (1 - \frac{1}{\chi})n$, there are at least $(1 - \frac{1}{\chi})n$ edges incident on $C_{\max}$.

Now the vertices outside $C_{\max}$ are colored with $(\chi - 1)$ colors. Thus there must exists a C_i such that at least $\frac{(1-\frac{1}{\chi})n}{\chi-1} = n/\chi$ edges incident on $C_{\max}$ are also incident on C_i. Now the adversary answers all the vertices in that C_i to be present. Then one must check at least n/χ vertices from $C_{\max}$ because as soon as any one of them is present we have an edge in the graph. □

6.4 Results on Restricted Graph Classes

Recall that, our setting provides a platform to compare the complexity of $\mathcal{P}$ when the base graph G has a certain property or structure. Recall that, the notion of $\mathcal{G}$-min-$cost(\mathcal{P})$ for a class of graphs $\mathcal{G}$ is obtained by restricting ourselves only to graphs in $\mathcal{G}$.

$$\mathcal{G}\text{-}min\text{-}cost_n(\mathcal{P}) = \min_{G \in \mathcal{G}}\{cost(\mathcal{P}, G) \mid G \text{ is relevant for } \mathcal{P} \ \& \ |V(G)| = n\}.$$

Below we consider the class of planar graphs and prove several properties in the node query settings. Note that it is always possible to consider other class of graphs and study several relevant properties.

Independence/Emptiness in Planar Graph

Theorem 6.4.1. *Let $\mathcal{G}_{\mathcal{P}}$ be a family of planar graphs on n vertices and $\mathcal{I}$ denote the property of being independent. Then,*

$$\mathcal{G}_{\mathcal{P}}\text{-}min\text{-}cost(\mathcal{I}) = \Omega(n).$$

Proof. As planar graphs are 4 colorable using Theorem 6.3.13 we can directly conclude this theorem. □

6.4.1 Triangle-Freeness in Planar Graphs

A graph G is called *inherently sparse* if every subgraph of G on k nodes contains $O(k)$ edges.

Theorem 6.4.2. *Let $\mathcal{G}_s$ be a family of inherently sparse graphs on n vertices and $\mathcal{T}$ denote the property of being triangle-free. Then,*

$$\mathcal{G}_s\text{-}min\text{-}cost(\mathcal{T}) = \Omega(\sqrt{n}).$$

Proof. Let $G = (V, E) \in \mathcal{G}_s$ be a graph on n vertices such that every vertex in G is part of at least one triangle in G. Let $S \subseteq V$ be a minimal hitting set for triangles in G. Now consider the following two cases.

Case 1: $|S| \geq \sqrt{n}$ The adversary answers all the vertices outside S to be present. Hence one has to check all vertices in S.

Case 2: $|S| < \sqrt{n}$ Since G is inherently sparse, there can be at most $O(|S|) = O(\sqrt{n})$ edges

within S. Moreover each triangle in G must share either one vertex in S or one edge inside S (it could even be that the whole triangle is inside S). Note that there are $\Omega(n)$ vertices outside S and each of them must belong to some triangle. This implies that either there is at least a vertex $v \in S$ or at least an edge $\{u, v\} \in \binom{S}{2}$ s.t. v or $\{u, v\}$ supports at least $\Omega(\sqrt{n})$ triangles outside S. Otherwise all triangles in G are not covered by S. We consider the following two cases:

Case 2a: an edge $\{u, v\} \in \binom{S}{2}$ supports $\Omega(\sqrt{n})$ triangles in $V - S$.

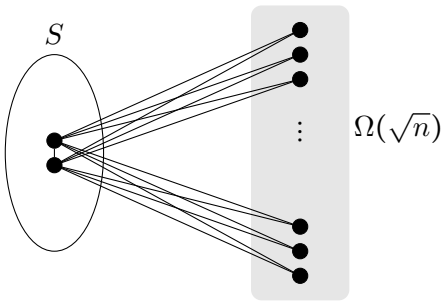


Figure 6.4: Case 2a of Triangle-Freeness in Planar Graphs

The adversary makes the edge $\{u, v\}$ present. One then has to query $\Omega(\sqrt{n})$ end points of all $\Omega(\sqrt{n})$ triangles (see Figure 6.4).

Case 2b: a vertex $v \in S$ supports $\Omega(\sqrt{n})$ triangles.

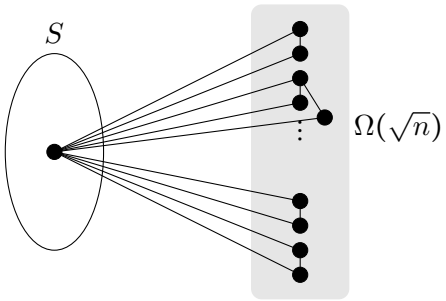


Figure 6.5: Case 2b of Triangle-Freeness in Planar Graphs

Adversary makes the vertex v present. Then the problem reduces to finding an edge in the graph induced on neighbors of v. This graph has $\Omega(\sqrt{n})$ non-isolated vertices. (see Figure 6.5). As the G is inherently sparse the chromatic number of G is constant. The idea is to pick a smallest degree vertex and recursively color the rest of the graph with constant colors. Use a different color for the picked vertex. Now we use Theorem 6.3.13 to obtain the $\Omega(\sqrt{n})$ bound. □

As a consequence of Lemma 6.4.2 we get the following:

Corollary 6.4.3. *Let $\mathcal{G}_{\mathcal{P}}$ be a family of planar graphs on n vertices and $\mathcal{T}$ denote the property of being triangle-free. Then,*

$$\mathcal{G}_{\mathcal{P}}\text{-}min\text{-}cost(\mathcal{T}) = \Omega(\sqrt{n}).$$

6.4.2 Acyclicity in Planar Graphs

We prove the bounds for acyclicity in 3-connected planar graphs. A connected graph G is called k-connected if the graph G remains connected even after deleting any $k-1$ vertices from the graph.

Theorem 6.4.4. *Let $\mathcal{G}_{\mathcal{P}_3}$ be a family of 3-connected planar graphs and $\mathcal{C}$ denote the property of being acyclic. Then,*

$$\mathcal{G}_{\mathcal{P}_3} - min\text{-}cost(\mathcal{C}) = \Omega(\sqrt{n}).$$

Proof. Let $G \in \mathcal{G}_{\mathcal{P}_3}$ be a graph on n vertices and m edges such that every vertex is relevant for the acyclicity property. Let d_{max} denote the maximum degree of G.

Case 1: $d_{\max} > \sqrt{n}$: We use the following fact: In 3-connected planar graphs, removing any vertex leaves a facial cycle around it. We apply this for the maximum degree vertex. In other words, we have a (not necessarily induced) wheel with d_{max} spokes (some spokes might be missing). See Figure 6.6. The adversary answers the central vertex of the wheel to be present. We can find a matching of size $\Omega(\sqrt{n})$ among the vertices of the cycle. Hence we have $\Omega(\sqrt{n})$ sensitive blocks of length 2 each, which can not be left un-queried.

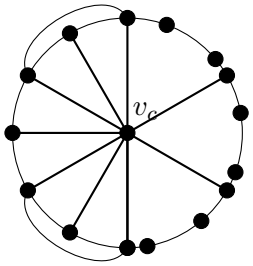


Figure 6.6: A Wheel with d_{max} Spokes

Case 2: $d_{\max} \leq \sqrt{n}$: We use the fact that every 3-connected graph must have at least $3n/2$ edges. Now using Lemma 6.2.5 we obtain a lower bound of $(m-n)/d_{max} \geq \Omega(\sqrt{n})$.

□

We believe that with similar arguments the above proof can be extended to general planar graphs.

6.5 Non-Hereditary Properties

So far we have studied hereditary properties and have seen that in the present of minimum symmetry assumption on the base graph the complexity remains evasive, whereas on the other hand when we take away any symmetry assumptions from the base graph the complexity falls down at least quadratically. In this section we study non-hereditary properties and show that they behave significantly differently.

6.5.1 Global vs. Local Connectivity

Theorem 6.5.1. *Let $Global\text{-}Con$ denote the problem of testing whether a graph is connected or not and let $Local\text{-}Con$ denote the problem of testing given two specified vertices s and t whether there is a path between s and t. Then,*
(a) $min\text{-}cost(Local\text{-}Con) = \Theta(1)$ *whereas*
(b) $min\text{-}cost(Global\text{-}Con) = \Theta(n)$.

Proof. This theorem follows directly from Lemma 6.5.2 and Lemma 6.5.3. □

Lemma 6.5.2. *Let $Local\text{-}Con$ denote the problem of testing given two specified vertices s and t whether there is a path between s and t. Then,*

$$min\text{-}cost(Local\text{-}Con) = O(1).$$

Proof. Consider the star graph on n vertices, i.e., a vertex v connected to $n-1$ vertices $u_1, \ldots, u_{n-1}$. It is easy to check that for any s and t we have to check at most one more vertex other than s and t to determine if there is a path between s and t. □

Lemma 6.5.3. *Let $Global\text{-}Con$ denote the problem of testing whether a graph is connected or not. Then,*

$$min\text{-}cost(Global\text{-}Con) = \Omega(n).$$

Proof. The convention we use is that the singleton vertex is connected and graph on 0 nodes is connected. Let G be a graph on n vertices. We are interested in global connectivity of subgraphs

of G. If the complement of G has a matching of size $\Omega(n)$ then each matching edge is a sensitive block with respect to empty graph, i.e., the graph with just these two nodes present is not connected. Hence we get the desired lower bound. If on the other hand the maximum matching size in the complement of G is at most say $n/4$ (hence the vertex cover size at most $n/2$), then G must contain a clique on at least $n/2$ vertices.

Now we consider vertices outside this clique. Since the connectivity property is non-trivial on G, at least one such vertex say v must exist. If v has at most $n/4$ edges to the vertices of the clique, we answer this vertex to be present and its neighbors in clique to be absent. Now as soon as any of the remaining $n/4$ vertices from the clique are present, we get a disconnected graph. If v has at least $n/4$ neighbors in the clique, we take a non-neighbour say u from the clique. Now u and v have at least $n/4$ common neighbors. We make u and v to be present. As soon as any of their common neighbors are present the graph is connected. This gives us $\Omega(n)$ bound. □

6.5.2 Perfect Matching

Proof of the following theorem is similar to the proof of Theorem 6.5.1 and can be derived easily.

Theorem 6.5.4. *Let $\mathcal{PM}$ be the property of containing a perfect matching in a graph G on n vertices then,*

$$\textit{min-cost}(\mathcal{PM}) = \Theta(n).$$

Chapter 7

The Power of Quantum Messages and Quantum Proofs in Communication Protocols

This chapter is based on the following paper:

- *Klauck, H., and Podder, S. (2014). Two results about quantum messages. In Proceedings of the 39th International Symposium on the Mathematical Foundations of Computer Science 2014. Springer Berlin Heidelberg, 2014. Pages 445-456. [KP14b]*

7.1 Introduction

The second part of the thesis is about communication complexity and communication protocols. In this chapter we try to understand the power of quantum proofs and quantum messages in the communication complexity setting. The power of using quantum messages over classical messages is a central topic in information and communication theory. It is always good to understand such questions well in the simplest settings where they arise. An example is the communication complexity model, which is rich enough to lead to many interesting results, yet accessible enough for us to show results about deep questions like the relationship between different computational modes, e.g. quantum versus classical or nondeterministic versus deterministic.

7.1.1 One-way Communication Complexity

Perhaps the simplest question one can ask about the power of quantum messages is the relationship between quantum and classical one-way protocols. Alice sends a message to Bob in order to compute the value of a function $f : \{0,1\}^n \times \{0,1\}^m \to \{0,1\}$. Essentially, Alice communicates a quantum state and Bob performs a measurement, both depending on their respective inputs. Though deceptively simple, this scenario is not at all fully understood. Let us just mention the following open problem: what is the largest complexity gap between quantum and classical protocols of this kind for computing a total Boolean function? The largest gap known is a factor of 2, as shown by Winter [Win04], but for all we know there could be examples where the gap is exponential, as it indeed is for certain partial functions [GKK+08].

An interesting bound on such speedups can be found by investigating the effect of replacing quantum by classical messages. Let us sketch the proof of such a result. Suppose a total Boolean function f has a quantum one-way protocol with communication c, namely Alice sends c qubits to Bob, who can decide f with error 1/3 by measuring Alice's message. We allow Alice and Bob to share an arbitrary input-independent entangled state. Extending Nayak's random access code bound [Nay99] Klauck [Kla00] showed that $Q^{A\to B,*}(f) \geq \Omega(VC(f))$, where $Q^{A\to B,*}(f)$ denotes the entanglement-assisted quantum one-way complexity of f, and $VC(f)$ the Vapnik-Chervonenkis dimension of the communication matrix of f. Together with Sauer's Lemma [Sau72] this implies that $D^{A\to B}(f) \leq O(Q^{A\to B,*}(f) \cdot m)$, where m is the length of Bob's input. See also [JZ09] for a related result where the authors get a bound of $D_\epsilon^{A\to B,\mu}(f) = O((I(X:Y)+1) \cdot VC(f))$, for non-product distributions μ on $\mathcal{X} \times \mathcal{Y}$.

A result such as the above is much more interesting in the case of partial functions. The reason is that for total functions a slightly weaker statement follows from a weak version of the random access code bound, which can be (and indeed has been [ANTV99]) established by the following argument: boost the quantum protocol for f until the error is below 2^{-2m}, where m is Bob's input length. Measure the message sent by Alice with *all* the measurements corresponding to Bob's inputs (this can be done with small total error) in order to determine Alice's row of the communication matrix and hence her input. This is a hard task by standard information theory facts (Holevo's bound). When considering partial functions a disaster happens: Bob does not know for which of his inputs y the value $f(x,y)$ is defined. If Bob measures the message for x with the observable for y and $f(x,y)$ is undefined any acceptance probability is possible and the message

state can be destroyed.

Aaronson [Aar05] circumvented this problem in the following way: Bob now tries to learn Alice's message. He starts with a guess (the totally mixed state) and keeps a classical description of his guess. Alice also always knows what Bob's guess is. Bob can simulate quantum measurements by brute-force calculation: for any measurement operator Bob can simply calculate the result from his classical description. Alice can do the same. Since Bob has some 2^m measurements he is possibly interested in, Alice can just tell him on which of these he will be wrong. Bob can then adjust his quantum state accordingly, and Aaronson's main argument is that he does not have to do this too often before he reaches an approximation of the message state. Note that Bob might never learn the message state if it so happens that all measurements are approximately correct on his guess. But if he makes a certain number of adjustments he will learn the message state and no further adjustments are needed.

Let us state Aaronson's result from [Aar05].

Lemma 7.1.1. *$D^{A\to B}(f) \leq O(Q^{A\to B}(f)\cdot \log(Q^{A\to B}(f))\cdot m)$ for all partial Boolean $f : \{0,1\}^n \times \{0,1\}^m \to \{0,1\}$.*

Aaronson later proved the following result, that removes the log-factor at the expense of having randomized complexity on the left hand side [Aar07].

Lemma 7.1.2. *$R^{A\to B}(f) \leq O(Q^{A\to B}(f)\cdot m)$ for all partial Boolean $f : \{0,1\}^n \times \{0,1\}^m \to \{0,1\}$.*

The first result is the following improvement.

Theorem 7.1.3. *For all partial Boolean $f : \{0,1\}^n \times \{0,1\}^m \to \{0,1\}$,*

$$D^{A\to B}(f) \leq O(Q^{A\to B,*}(f)\cdot m).$$

Here $Q^{A\to B,*}(f)$ denotes the one-way quantum communication complexity of f where the players have shared entanglements.

Hence we remove the log-factor, and we allow the quantum communication complexity on the right hand side to feature prior entanglement between Alice and Bob. Arguably, looking into the entanglement assisted case (which is interesting for the second main result) led us to consider a more systematic progress measure than in Aaronson's proof, which in turn allowed us to analyse

a different update rule for Bob that also works for protocols with error 1/3, instead of extremely small error as used in [Aar05], which is the cause of the lost log-factor.

We note that this result can be used to slightly improve on the "quantum-classical" simultaneous message passing lower bound for the Equality function by Gavinsky et al. [GRdW08], establishing a tight $\Omega(\sqrt{n})$ lower bound on the complexity of Equality in a model where quantum Alice and classical Bob (who do not share a public coin or entanglement) each send messages to the referee. The tight lower bound has also recently been established via a completely different and simpler method [BGK15] (as well as generalized to a nondeterministic setting). Our result (as well as the one in [BGK15]) allows a generalization to a slightly stronger model: Alice and the referee may share entanglement.

7.2 Simulating One-way Quantum Messages Protocol Using Deterministic Messages

In this section we prove the Theorem 7.1.3. Our proof holds for partial functions. Note that in the case of total functions the theorem follows from a result in [Kla00] combined with Sauer's lemma [Sau72].

Our proof follows Aaronson's main approach in [Aar05], in which Bob maintains a classical description of a quantum state as his guess for the message he should have received, and Alice informs him about inputs on which this state will perform badly, so that he can adjust his guess. His goal is to either get all measurement results approximately right, or to learn the message state. We will refer to these states as the current guess state, and the target state.

We deviate from Aaronson's proof in two ways. First, we work with a different progress measure that is more transparent than Aaronson's, namely the relative entropy between the target state and the current guess. This already allows us to work in the entanglement-assisted case.

Secondly, we modify the rule by which Bob updates his guess. In Aaronson's proof Bob projects his guess state onto the subspace on which the target state has a large projection (because the message is accepted by the corresponding measurement with high probability). This has the drawback that one cannot use the actual message state of the protocol as the target state, because that state usually has considerable projection onto the orthogonal complement of the subspace, making the relative entropy infinitely large! Hence Aaronson uses a boosted and projected message state as the target state. This state is close to the actual message state thanks to the boosting, and

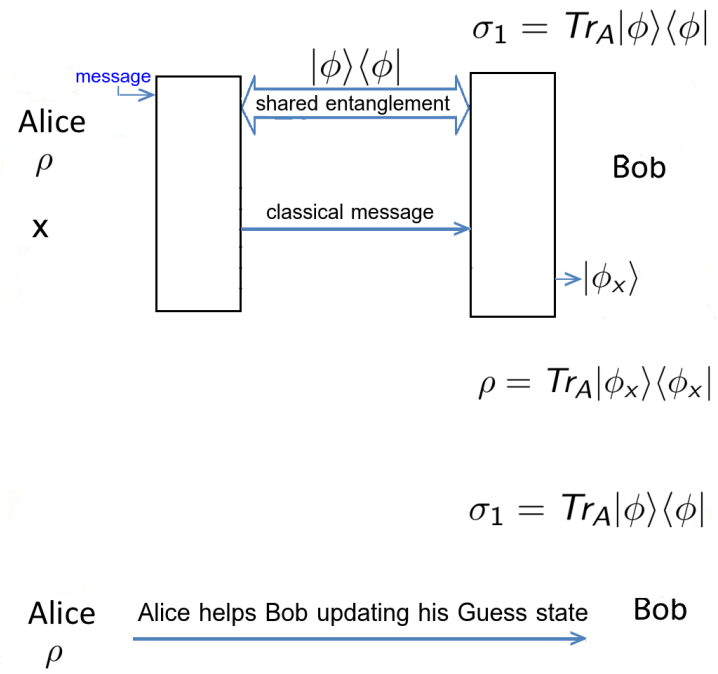


Figure 7.1: Making Quantum Message Deterministic

projection of the guess state now properly decreases the relative entropy, since the target state is fully inside the subspaces. The boosting step costs exactly the log-factor we aim to remove.

So in the situation where Bob wants to update his guess state σ, knowing that the target state ρ will be accepted with probability $1 - \epsilon$ when measuring the observable consisting of subspace V_y and its complement, we let Bob replace σ with the mixture of $1 - \epsilon$ times the projection onto V_y and ϵ times the projection onto $V_y^{\perp}$. The main part of the proof is then to show that this decreases the relative entropy $S(\rho||\sigma)$ given that $\mathrm{Tr}(V_y\sigma) < 1 - 10\sqrt{\epsilon}$, i.e., in case σ was not good enough already. Eventually either all measurements can be done by Bob giving the correct result, or the current guess state σ satisfies $S(\rho||\sigma) \leq 5\sqrt{\epsilon}$, in which case ρ and σ are also close in the trace distance meaning that any future measurement will give almost the same results on both states (see Figure 7.1).

Let us now give a more formal proof of the Theorem 7.1.3.

Proof. Fix any entanglement assisted one-way protocol with quantum communication $q = Q^{A\to B,*}(f)$. Using standard boosting we may assume that the error of the protocol is at most $\epsilon = 10^{-6}$ for any input x, y. This increases the communication by a constant factor at most.

Using teleportation we can replace the quantum communication by $2q$ classical bits of communication at the expense of adding q EPR-pairs to the shared entangled state. Let $|\phi\rangle$ denote the entangled state shared by the new protocol. We can assume this is a pure state, because if this is

not the case we may consider any purification, and Alice and Bob can ignore the purification part. Note that we do not restrict the number of qubits used in $|\phi\rangle$.

In the protocol, for a given input x Alice has to perform a unitary transformation on her part of $|\phi\rangle$ (we assume that any extra space used is also included in $|\phi\rangle$ and that measurements are replaced by unitaries) and then sends a classical message. Bob first applies the unitary from the teleportation protocol (which only depends on the classical message). Let's denote the state shared by Alice and Bob at this point by $|\phi_x\rangle$. Following this Bob performs a measurement (depending on his input y) on his part of $|\phi_x\rangle$. This measurement determines the output of the protocol on x, y. We may assume by standard techniques that Bob's measurements are projection measurements, and that the subspaces used in the projection measurements have dimension $d/2$, where d is the dimension of the underlying Hilbert space.

Recall that $|\phi\rangle$ and $|\phi_x\rangle$ are bipartite states shared by Alice and Bob. Let $\rho = \mathsf{Tr}_A|\phi_x\rangle\langle\phi_x|$ and $\sigma_1 = \mathsf{Tr}_A|\phi\rangle\langle\phi|$, i.e., the states when Alice's part is traced out. Bob wants to learn ρ in order to be able to determine all measurement results on ρ. We show how to do this with $O(m \cdot q)$ bits of deterministic communication from Alice. Note that the state σ_1 is known to Bob in the sense that he knows its classical description.

Since Alice's local operations do not change Bob's part of $|\phi\rangle\langle\phi|$, the difference between ρ and σ_1 is introduced via the correction operations in the teleportation protocol that Bob applies after he receives Alice's message. But with probability 2^{-2q} Bob does not have to do anything, i.e., when Alice's message is the all 0-s string. This implies that

$$\sigma_1 = \frac{1}{2^{2q}}\rho + \theta,$$

for some positive semidefinite θ with trace $1 - 1/2^{2q}$. Hence we get that

$$S(\rho\|\sigma_1) \leq S_\infty(\rho\|\sigma_1) \leq 2q.$$

In other words, Bobs target ρ and initial guess σ_1 have small relative entropy.

We can now describe the protocol. Bob starts with the classical description of σ_1. This state is also known by Alice, since it does not depend on the input. Throughout the protocol Bob will hold states σ_i, which will be updated when needed, using information provided by Alice. Bob also has a set of measurement operators $P_y, I - P_y$ for all his inputs y. Bob and Alice each loop over his inputs y, and compute $p_y = \mathsf{Tr}(P_y\sigma_i)$. This is the acceptance probability, if σ_i is

measured with the measurement for his input y. Alice also computes $p'_y = \mathsf{Tr}(P_y\rho)$ which is the acceptance probability of the quantum protocol. If p_y and p'_y are too far apart, Alice will notify Bob of the correct acceptance probability on y (with precision ϵ^2), which takes $m + O(1)$ bits of communication.

Alice does not send a message if $f(x, y)$ is undefined, because the acceptance probability on such inputs is irrelevant. Suppose $p'_y = 1 - \epsilon_y$, where $\epsilon_y \leq \epsilon$ is the error on x, y (and $f(x, y) = 1$), but $p_y = 1 - a$ for some $1 > a \geq 10\sqrt{\epsilon}$. If this is not the case the measurement for y applied to σ_i already yields the correct result and no information from Alice is needed. So if p_y, p'_y are far apart Alice will send y (using m bits) and ϵ_y as a floating point number with precision ϵ^2 (using $O(1)$ bits).

Bob then adjusts σ_i to obtain a state σ_{i+1}. Suppose he knows the correct ϵ_y (the difference between ϵ_y and its approximation sent by Alice will be irrelevant). This means that $\mathsf{Tr}(P_y\sigma_i) = 1-a$ but $\mathsf{Tr}(P_y\rho) = 1 - \epsilon_y$. P_y is the projector onto a subspace V_y. We have assumed that each V_y has dimension $d/2$ if d is the dimension of the underlying Hilbert space. Let B_i denote an orthonormal basis, in which the first $d/2$ elements span V_y, and the remaining $d/2$ span $V_y^\perp$. Furthermore in this basis the upper left and lower right quadrants of σ_i are diagonal. Hence σ_i will look like this:

$$\sigma_i = \begin{pmatrix} A & B \\ B^* & D \end{pmatrix} = \begin{pmatrix} * & 0 & 0 & * & \cdots & * \\ 0 & \ddots & 0 & * & \vdots & * \\ 0 & 0 & * & * & \cdots & * \\ * & \cdots & * & * & 0 & 0 \\ * & \vdots & * & 0 & \ddots & 0 \\ * & \cdots & * & 0 & 0 & * \end{pmatrix},$$

where $\mathsf{Tr}(A) = 1 - a$ and $\mathsf{Tr}(D) = a$. We can now define

$$\sigma_{i+1} = \begin{pmatrix} \frac{1-\epsilon_y}{1-a}A & 0 \\ 0 & \frac{\epsilon_y}{a}D \end{pmatrix}.$$

σ_{i+1} is diagonal in our basis B_i. Clearly σ_{i+1} would perform exactly as desired on measurement $P_y, I - P_y$. But Bob already knows the function value on y and can carry on with the next y.

Before we continue we have to argue that the case $a = 1$ can never happen. Since $\mathsf{Tr}(P_y\rho) = 1 - \epsilon_y > 0$ the state ρ has a nonzero projection onto V_y. If $a = 1$ then σ_i sits entirely in $V_y^\perp$, and hence $S(\rho||\sigma_i) = \infty$. But since we start with a finite $S(\rho||\sigma_1)$ and only decrease that value the

situation $a = 1$ is impossible.

Coming back to the protocol, it is obvious that Bob will learn the correct value of $f(x, y)$ for all y such that $f(x, y)$ is defined. Hence the protocol is deterministic and correct. The remaining question is how many times Alice has to send a message to Bob. We will show that this happens at most $O(Q^{A\to B,*}(f)/\sqrt{\epsilon})$ times, which establishes our theorem.

The main claim that remains to be shown is the following.

Claim 7.2.1. *$S(\rho||\sigma_i) \geq S(\rho||\sigma_{i+1}) + a/2$ if $a \geq 10\sqrt{\epsilon}$.*

This establishes the upper bound on the number of messages, because the relative entropy, which starts at $2q$ is decreased by $a/2 \geq 5\sqrt{\epsilon}$ for each message. After at most $2q/(5\sqrt{\epsilon})$ iterations the protocol has either ended (in which case Bob might never learn ρ, but will still know all measurement results), or we have

$$S(\rho||\sigma_T) \leq 5\sqrt{\epsilon}.$$

To see this assume we are still in the situation of the claim. The claim states that the relative entropy can be reduced by $a/2$ as long as $a \geq 10\sqrt{\epsilon}$. So the process stops (assuming we don't run out of suitable $y's$) no earlier than when $S(\rho||\sigma_i) < a/2 \leq 5\sqrt{\epsilon}$.

But then we can use the quantum Pinsker inequality [HOT81], see also [KNTSZ01].

Fact 7.2.2.

$$||\rho - \sigma||_t \leq \sqrt{2 \ln 2 \; S(\rho||\sigma)}.$$

Thus we have that at the final time T: $||\rho - \sigma_T||_t \leq \sqrt{10 \ln 2\sqrt{\epsilon}} < 0.1$ in the end, and hence for *all* measurements their results are close. Hence no more than $O(q/\sqrt{\epsilon}) = O(q)$ messages have to be sent.

We now finish the proof by showing the claim.

Define another state

$$\tilde{\sigma_i} = \begin{pmatrix} A & 0 \\ 0 & D \end{pmatrix},$$

i.e., the state σ_i with its upper right and lower left quadrants deleted. It is easy to see that this is still a density matrix. Indeed $\tilde{\sigma_i}$ is the state σ_i after measuring $P_y, I - P_y$. Also define $\tilde{\rho}$ to be the matrix ρ with its upper left and lower right quadrants replaced by 0, which is again a density

matrix resulting from measuring ρ. If

$$\rho = \begin{pmatrix} E & F \\ G & H \end{pmatrix} \text{ then } \tilde{\rho} = \begin{pmatrix} E & 0 \\ 0 & H \end{pmatrix}.$$

We use Uhlmann monotonicity.

Fact 7.2.3. *If $\tilde{\rho}, \tilde{\sigma}$ result from measuring ρ, σ then*

$$S(\tilde{\rho}||\tilde{\sigma}) \leq S(\rho||\sigma).$$

By Uhlmann monotonicity we have $S(\rho||\sigma_i) \geq S(\tilde{\rho}||\tilde{\sigma_i})$, and it suffices to show that $S(\tilde{\rho}||\tilde{\sigma_i}) \geq S(\rho||\sigma_{i+1}) + a/2$. Note that both $\tilde{\sigma_i}$ and σ_{i+1} are diagonal in the basis B_i. Furthermore $S(\rho||\sigma_{i+1}) = S(\tilde{\rho}||\sigma_{i+1}) + S(\tilde{\rho}) - S(\rho)$.

We first bound the term $S(\tilde{\rho}) - S(\rho)$. In the basis B_i we may view ρ as a bipartite state ρ_{RQ} consisting of a qubit R (corresponding to membership in V_y) and the remaining qubits Q. In $\tilde{\rho}$ the qubit R has been measured. Consider attaching another qubit T, and instead of measuring R applying the unitary that "copies" R to T. After the unitary $S(\rho) = S(\rho_{QRT}) \geq S(\rho_{QR}) - S(\rho_T) = S(\tilde{\rho}) - S(\rho_T)$, due to the Araki-Lieb inequality and because $\rho_{QR} = \tilde{\rho}$. But $S(\rho_T) = H(\epsilon_y)$ and hence $S(\tilde{\rho}) - S(\rho) \leq H(\epsilon_y)$.

We need to compare $\mathsf{Tr}(\tilde{\rho} \log \tilde{\sigma_i})$ and $\mathsf{Tr}(\tilde{\rho} \log \sigma_{i+1})$. Note that $\tilde{\sigma_i}$ and σ_{i+1} are both diagonal.

$$\begin{aligned}
& \mathsf{Tr}(\tilde{\rho} \log \tilde{\sigma_i}) \\
= & \sum_j \tilde{\rho}(j,j) \log(\tilde{\sigma_i}(j,j)) \\
= & \sum_{j \leq d/2} \tilde{\rho}(j,j) \log(\sigma_{i+1}(j,j) \cdot \frac{1-a}{1-\epsilon_y}) \\
+ & \sum_{j > d/2} \tilde{\rho}(j,j) \log(\sigma_{i+1}(j,j) \cdot \frac{a}{\epsilon_y}) \\
= & \sum_j \tilde{\rho}(j,j) \log(\sigma_{i+1}(j,j)) + (1-\epsilon_y) \cdot \log \frac{1-a}{1-\epsilon_y} + \epsilon_y \cdot \log \frac{a}{\epsilon_y} \\
= & \mathsf{Tr}(\tilde{\rho} \log \sigma_{i+1}) + H(\epsilon_y) - H(\epsilon_y, a),
\end{aligned}$$

where $H(u,v) = -u \log v - (1-u) \log(1-v)$.

Assuming that $a \geq 10\sqrt{\epsilon}$ and $\epsilon_y \leq \epsilon = 10^{-6}$ we can estimate

$$H(\epsilon_y, a) \geq a. \tag{7.1}$$

If $a > 1/2$ then (7.1) is true from the term $(1-\epsilon_y)\log(1/(1-a)) \geq a$. Otherwise for all $a \in [0,1]$ we have $-\log(1-a) \leq a$, hence the term in $H(\epsilon_y, a)$ is at least $(1-\epsilon_y)a$. And for the first term $\epsilon_y \log(1/a) \geq \epsilon_y \log(2) \geq \epsilon_y$, so (7.1) is always true.

Also note that $H(\epsilon_y) \leq H(\epsilon) = \epsilon \log(\epsilon^{-1}) \leq \sqrt{\epsilon} \leq a/10$.

But then

$$\begin{aligned}
& S(\rho\|\sigma_i) \\
\geq\ & S(\tilde{\rho}\|\tilde{\sigma}_i) \\
=\ & -S(\tilde{\rho}) - \mathsf{Tr}(\tilde{\rho}\log\tilde{\sigma}_i) \\
\geq\ & -S(\rho) - H(\epsilon) - \mathsf{Tr}(\tilde{\rho}\log\tilde{\sigma}_i) \\
\geq\ & -S(\rho) - H(\epsilon) - \mathsf{Tr}(\tilde{\rho}\log\sigma_{i+1}) + a - H(\epsilon) \\
=\ & -S(\rho) - \mathsf{Tr}(\rho\log\sigma_{i+1}) + a - 2H(\epsilon) \\
\geq\ & S(\rho\|\sigma_{i+1}) + a/2.
\end{aligned}$$

□

7.3 Quantum versus Classical Proofs

We now turn to the second result of this chapter, which is philosophically very interesting. Interactive proof systems are a fundamental concept in computer science. Quantum proofs have a number of disadvantages: reading them may destroy them, errors may occur during verification, verification needs some sort of quantum machine, and it may be much harder to provide them than classical proofs. The main hope is that quantum proofs can in some situations be verified using fewer resources than classical proofs. Until now such a hope has not been verified formally. In the fully interactive setting Jain et al. [JJUW11] have shown that the set of languages recognizable in polynomial time with the help of a quantum prover is equal to the set where the prover and verifier are classical (i.e., IP=QIP [JJUW11]).

The question remains open in the noninteractive setting. A question first asked by Aharonov

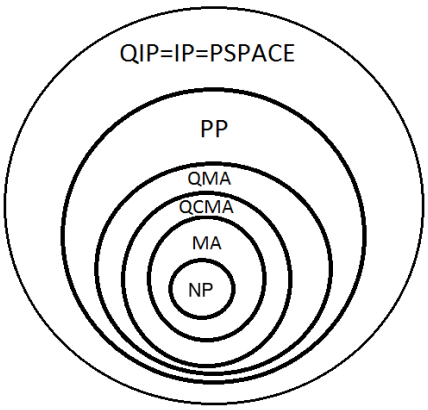


Figure 7.2: The Class QMA and $QCMA$

and Naveh [AN02] and meriting much attention, is whether proofs that are quantum states can ever be easier to verify than classical proofs (by quantum machines) in the absence of interaction, i.e., whether the class QMA is larger than its analogue with classical proofs but quantum verifiers, known as QCMA (see Figure 7.2 for the contentment of the classes QMA and QCMA in other complexity classes). An indication that quantum proofs may be powerful was given by Watrous [Wat00], who described an efficient QMA black box algorithm for deciding nonmembership in a subgroup. However, Aaronson and Kuperberg [AK07] later showed how to solve the same problem efficiently using a classical witness, giving a QCMA black box algorithm for the problem. They also introduced a quantum problem, for which they show that QMA black box algorithms are more efficient than QCMA black box algorithms. Using a quantum problem to show hardness for algorithms using classical proofs seems unfair though and a similar separation has remained open for Boolean problems.

In our second main result of this chapter we compare the two modes of noninteractive proofs and quantum verification for a Boolean function in the setting of one-way communication complexity. More precisely we exhibit a partial Boolean function f, such that the following holds. f can be computed in a protocol where a prover who knows x, y can provide a quantum proof to Alice, and Alice sends quantum message to Bob, such that the total message length (proof plus message Alice to Bob) is $O(\log n)$. In the setting where a prover Merlin (still knowing all inputs) sends a classical proof to Alice, who sends a quantum message to Bob, the total communication is $\Omega(\sqrt{n}/\log n)$.

Result 7.3.1. *There is a partial Boolean function f such that $QMA^{A\to B}(f) = O(\log n)$, while $QCMA^{A\to B,*}(f) = \Omega(\sqrt{n}/\log n)$.*

We note that this is the first known exponential gap between computing Boolean functions in a

QCMA and a QMA mode in any model of computation. Also, the lower bound is not too far from being tight, since there is an obvious upper bound of $O(\sqrt{n}\log n)$ for the problem.

So where does the power of quantum proofs come from in our result? Raz and Shpilka [RS04] show that QMA one-way protocols are as powerful as QMA two-way protocols. Their proof uses a quantum witness that is a superposition over the messages of different rounds. We show that for a certain problem with an efficient QMA protocol there is no efficient one-way QCMA protocol. Hence the weakness of classical proofs here is the impossibility of compressing interaction as in the QMA case.

Let us first define the problem for which we prove our separation result.

Definition 7.3.2. *The function* $\mathsf{MajIx}(x, I)$*, where* $I = \{i_1, \ldots, i_{\sqrt{n}}\}$*, each* $i_j \in \{1, \ldots, n\}$*, and* $x \in \{0,1\}^n$ *is defined as follows:*

1. *if* $|\{j : x_{i_j} = 1\}| = \sqrt{n}$ *then* $\mathsf{MajIx}(x, I) = 1$,
2. *if* $|\{j : x_{i_j} = 1\}| \leq 0.9\sqrt{n}$ *then* $\mathsf{MajIx}(x, I) = 0$,
3. *otherwise* $\mathsf{MajIx}(x, I)$ *is undefined.*

The function has been studied in [Kla11], where it is shown that one-way MA protocols for the problem need communication $\Omega(\sqrt{n})$. Our main technical result here is to extend this to one-way QCMA protocols.

It is obvious on the other hand, that there is a cheap protocol when Bob can send a message to Alice.

Lemma 7.3.3. $R^{B\to A}(\mathsf{MajIx}) = O(\log n)$.

Raz and Shpilka [RS04] show that any problem with $QMA(f) = c$ (i.e., QMA protocol where Alice and Bob can interact over many rounds) has a QMA protocol of cost $poly(c)$ in which Merlin sends a message to Alice, who sends a message to Bob. By inspection of their proof the polynomial overhead can be removed in the case of constant rounds of interaction between Alice and Bob.

Lemma 7.3.4. *If* $QMA(f) = c$ *and this cost can be achieved by a protocol with* $O(1)$ *rounds, then* $QMA^{A\to B}(f) = O(c)$.

Proof. The proof of Raz and Shpilka in [RS04] proceeds by showing that a problem LSD is complete for QMA communication complexity. It is easy to see that LSD can be computed by a QMA one-way protocol with logarithmic communication. The main difficulty is showing that

any problem f with $QMA(f) = C$ can be reduced to an instance of LSD of size $2^{poly(C)}$. Here we will argue that if the QMA complexity of f is C for protocols that have only $O(1)$ rounds between Alice and Bob, then the constructed LSD instance has size $2^{O(C)}$ only, and hence we get that $QMA^{A\to B}(f) = O(C)$.

For the problem LSD (linear space distance), Alice and Bob each receive a subspace of $\mathbb{R}^n$ of dimension $n/4$. The distance between two subspaces V, W is the minimum over all pairs of unit vectors from V and W of the Euclidean distance between the vectors. For LSD the promise is that the spaces given to Alice and Bob are either very close (distance at most $0.1\sqrt{2}$) or very far (distance at least $0.9\sqrt{2}$). The 1-inputs of LSD are the pairs of spaces that are close. Note that there is a QMA one-way protocol for this problem of logarithmic cost: Merlin sends Alice a vector that is close to both subspaces as a quantum state, Alice measures the state with the observable consisting of her subspace and its complement, and if the state projects into her subspace she sends the projected state to Bob (otherwise she rejects), who measures with his observable. If both measurements succeed, they accept.

The main lemma in the reduction of [RS04] is the following (their Lemma 19):

Lemma 7.3.5. *If f has a QMA protocol with proof length p, communication c, and r rounds and error $1/r^4$, then f can be reduced (by local operation by Alice and Bob) to an instance of LSD where the underlying space is $\mathbb{R}^{(r+1)2^{2(c+p)}}$, and the distance between the two spaces is at least $1/r^{1.5}$ in the case of 0-inputs, and at most $\sqrt{2}/r^{2.5}$ in the case of 1-inputs.*

This lemma is then combined with standard boosting ideas to improve the gap in the distance, but for us the fact that amplifying the success probability by parallel repetition in a QMA one-way protocol is possible suffices. Furthermore, in the case of $r = O(1)$, the LSD instance provided by the above lemma already has a constant gap and the correct size of $2^{O(c+p)}$. Reducing a function f to the LSD instance, and then running $O(1)$ times the protocol for LSD in parallel suffices to get a one-way QMA-protocol for f.

□

The Lemma 7.3.4 immediately implies the following.

Theorem 7.3.6. $QMA^{A\to B}(\mathsf{MajIx}) = O(\log n)$.

We now give a self-contained proof of this fact. Our protocol (see Figure 7.3) has completeness 1, hence even the one-sided error version of $QMA^{A\to B}$ is separated from $QCMA^{A\to B}$ by the following lower bound.

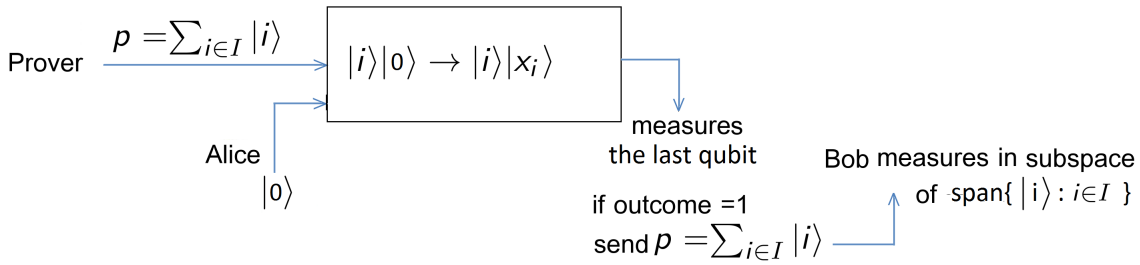


Figure 7.3: QMA One-way Protocol for MajIx

Proof. We describe a protocol with constant gap, which can be amplified to have the usual soundness using standard techniques.

Alice holds a string $x \in \{0,1\}^n$, Bob a set I of $\sqrt{n}$ distinct indices. Alice receives a proof from Merlin, which is supposed to be the uniform superposition $\sum_{i \in I} |i\rangle$. Alice attaches one more qubit, applies the unitary that maps $|i\rangle|a\rangle$ to $|i\rangle|x_i \oplus a\rangle$, then measures that extra qubit. If the result is 0 she rejects. Otherwise she discards the extra qubit and sends the remaining state to Bob. Bob measures this state with the observable that consists of the subspace spanned by the vector with 1 in all positions $i \in I$ and 0 elsewhere, and its orthogonal complement. He accepts, if this measurement projects onto the first subspace. Note that the communication is $\log n$ qubits from both Merlin and Alice.

If Merlin is honest (and $x_i = 1$ for all $i \in I$), he sends the uniform superposition $\sum_{i \in I} |i\rangle$. Alice's measurement will not change this state (she effectively projects into the space spanned by the $|i\rangle$ with $x_i = 1$). Bob's measurement will accept with certainty. Hence the protocol has completeness 1.

Now assume that at most $0.9\sqrt{n}$ of the $i \in I$ satisfy that $x_i = 1$. We may assume that Merlin's message is a pure state of the correct dimension. For each mixed state there is a pure state that will perform at least as well, and states with the wrong dimension will be rejected immediately. Again, unless Alice rejects, her measurement projects into the space spanned by $|i\rangle$ for which $x_i = 1$. Denote this state by $|\psi\rangle = \sum_{i:x_i=1} \alpha_i |i\rangle$. Bob measures the observable consisting of span$(|\phi_I\rangle)$, where $|\phi_I\rangle = \sum_{i \in I} |i\rangle / n^{1/4}$, and its orthogonal complement. The probability of the measurement accepting is the squared inner product of $|\phi_I\rangle$ and $|\psi\rangle$. This value is $(\sum_{i \in I: x_i=1} \alpha_i / n^{1/4})^2 \leq (1/\sqrt{n})(0.9\sqrt{n})(\sum |\alpha_i|^2) \leq 0.9$. Hence the protocol has soundness error 0.9, which can be improved by parallel repetition. □

Theorem 7.3.7. $QCMA^{A \rightarrow B,*}(\mathsf{MajIx}) \geq \Omega(\sqrt{n}/\log n)$.

Hence we can conclude the following.

Corollary 7.3.8. *There is a partial Boolean function f such that $QMA^{A\to B}(f) = O(\log n)$, while $QCMA^{A\to B,*}(f) = \Omega(\sqrt{n}/\log n)$.*

Proof. Fix any QCMA protocol $\mathcal{P}$ for Majlx. Furthermore define a distribution on inputs as follows. Fix any error correcting code $C \subseteq \{0,1\}^n$ with distance $n/4$ (i.e., every two codewords have Hamming distance at least $n/4$). Such codes of size $2^{\Omega(n)}$ exist by the Gilbert-Varshamov bound. We do not care about the complexity of decoding and encoding for our code. Furthermore we require the code to be balanced, i.e., that any codeword has exactly $n/2$ ones. This can also be achieved within the stated size bound. For our distribution on inputs first choose $x \in C$ uniformly, and then uniformly choose I among all subsets of $\{1,\dots,n\}$ of size $\sqrt{n}$. Note that the probability of 1-inputs under the distribution μ just defined is between $2^{-\sqrt{n}}$ and $2^{-\sqrt{n}-1}$ due to the balance condition on the code.

If the cost (i.e., communication from Merlin plus communication from Alice) of $\mathcal{P}$ is c, then there are at most 2^c different classical proofs sent by Merlin. We identify such proofs p with the set of 1-inputs that are accepted by the protocol with probability at least $2/3$ when using the proof p. Hence there must be a proof P containing 1-inputs of measure at least $2^{-\sqrt{n}-c-1}$, because for every 1-input there is a proof with which it is accepted with probability $2/3$ or more. Furthermore, given P, no 0-input is accepted with probability larger than $1/3$. Note that inputs outside of the promise, or 1-inputs outside of P can be accepted with any probability between 0 and 1. Denote by f_P the partial function $\{0,1\}^n \times \{0,1\}^m \to \{0,1,\perp\}$, in which all inputs in P are accepted, and all 0-inputs of f are rejected, and the remaining inputs have undefined function value ($\perp$). $m = \Theta(\sqrt{n}\log n)$ is the length of Bob's input. Denote by $M = \binom{n}{\sqrt{n}}$ the number of Bob's inputs.

Obviously f_P can be computed by a one-way quantum protocol without prover using communication c (and possibly using shared entanglement between Alice and Bob). Now due to Theorem 7.1.3 this implies that $D^{A\to B}(f_P) \le O(c \cdot m)$. We will argue that on the other hand $D^{A\to B}(f_P) \ge \Omega(n)$, and hence $c \ge \Omega(n/m) = \Omega(\sqrt{n}/\log n)$, which is our theorem.

Denote by A the communication matrix of f_P. A row of A is *fat*, if it contains more than $M2^{-\sqrt{n}-2c}$ 1-inputs (to f_P). Note that there can be no fewer than $|C|2^{-c-2}$ fat rows, because there are at least $|C|M2^{-\sqrt{n}-c-1}$ 1-inputs in P, the non-fat rows contain together at most $|C|M2^{-\sqrt{n}-2c}$ 1-inputs and each fat row at most $M2^{-\sqrt{n}}$ 1-inputs. Let C' denote the row set consisting of the fat rows only, and A' the matrix A restricted to those rows. We claim that A' has C' distinct rows.

Recall that for distinct rows there is a column, where one row has a 1 entry and the other a 0 entry. This means that $D^{A\to B}(f_P) \geq \log C' \geq \log|C| - c - 2 \geq \Omega(n)$ (unless $c = \Omega(n)$ already).

To show that all pairs of rows in A' are distinct consider two of them, named x, y. We identify the row labelled by x with the set for which x is the characteristic vector. Recall x, y are both codewords and are both fat. $x \cap y \leq n(1/2 - 1/8)$ because x and y have Hamming distance at least $n/4$. Let $S \subseteq \mathcal{I}$ be the set of I such that $(x, I) \in P$. Then for all $I \in S$ we have that all of the $i \in I$ must satisfy $x_i = 1$.

Furthermore let $T \subseteq S$ be the set $I \in S$, such that $|I \cap x \cap y| \geq 0.9\sqrt{n}$. Then

$$|T| \leq \sqrt{n} \cdot \binom{\frac{3n}{8}}{0.9\sqrt{n}} \cdot \binom{\frac{n}{8}}{0.1\sqrt{n}} \leq 2^{-\alpha\sqrt{n}} \cdot \binom{\frac{n}{2}}{\sqrt{n}},$$

for some constant $\alpha > 0$. Note that the binomial coefficient on the right hand side is the number of $I \in \mathcal{I}$ such that (x, I) is a 1-input. Hence

$$\frac{\mu((\{x\} \times T) \cap P)}{\sum_{I \in \mathcal{I}:\mathsf{MajIx}(x,I)=1} \mu(x, I)} \leq 2^{-\alpha\sqrt{n}},$$

i.e., a small fraction of 1-inputs (x, I) in P on row x have $I \in T$, but x is fat and has more 1-inputs. We can assume that $c < \alpha\sqrt{n}/10$ and hence $2^{-2c} \geq 2 \cdot 2^{-\alpha\sqrt{n}}$, so that the set T contributes little to the set of $I \in \mathcal{I}$ with $(x, I) \in P$. In particular there must be at least one $I \notin T$ such that (x, I) is in P, and hence $f_P(x, I) = 1$.

So let us examine the set of all $I \in S - T$ (so $|I \cap x \cap y| < 0.9\sqrt{n}$). $|I \cap x| = \sqrt{n}$ and hence $|y \cap I| \leq .9\sqrt{n}$, i.e., (y, I) is a 0-input. This gives us the desired column I, such that $f_P(x, I) = 1$ and $f_P(y, I) = 0$, i.e., x, y are two distinct rows in A'.

□

Chapter 8

Garden-Hose Protocols

This chapter is based on the following paper:

- *Klauck, H., and Podder, S. (2014). New Bounds for the Garden-Hose Model. Foundation of Software Technology and Theoretical Computer Science. 2014, Page 481. LIPIcs-Leibniz International Proceedings in Informatics. Vol. 29. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2014. [KP14a]*

8.1 Introduction

In this chapter we shall investigate how computation is affected if we restrict ourself to a certain class of communication protocols – in particular we focus on the recently proposed garden-hose model [BFSS13] which is a memoryless and reversible communication model. We shall see that this model has beautiful connections with several other computational models and can be used to derive new lower bounds for other computational models. Investigation of this model will also broaden our understanding of time and space complexity and their trade-offs.

8.1.1 Background: The Garden-Hose Model

Recently, Buhrman et al. [BFSS13] proposed a new measure of complexity for finite Boolean functions, called *garden-hose complexity*. This measure can be viewed as a type of distributed space complexity, and while its motivation is mainly in applications to position based quantum cryptography, the playful definition of the model is quite appealing in itself. Garden-hose complexity can be viewed as a natural measure of space, in a situation where two players with private inputs compute a Boolean function cooperatively. Space-bounded communication complexity has

been investigated before [BTY94, Kla04, KŠdW07] (usually for problems with many outputs), and recently Brody et al. [BCP+13] have studied a related model of space bounded communication complexity for Boolean functions (see also [PSS14]). In this context the garden-hose model can be viewed as a memoryless model of communication that is also reversible.

To describe the garden-hose model let us consider two neighbors, Alice and Bob. They own adjacent gardens which happen to have s empty water pipes crossing their common boundary. These pipes are the only means of communication available to the two. Their goal is to compute a Boolean function on a pair of private inputs, using water and the pipes across their gardens as a means of communication. It is worth mentioning that even though Alice and Bob choose to not communicate in any other way, their intentions are not hostile and neither will deviate from a previously agreed upon protocol.

A garden-hose protocol works as follows: There are s shared pipes. Alice takes some pieces of hose and connects pairs of the open ends of the s pipes. She may keep some of the ends open. Bob acts in the same way for his end of the pipes. The connections Alice and Bob place depend on their local inputs x, y, and we stress that every end of a pipe is only connected to at most one other end of a pipe (meaning no Y-shaped pieces of hose may be used to split or combine flows of water). Finally, Alice connects a water tap to one of those open ends on her side and starts the water. Based on the connections of Alice and Bob, water flows back and forth through the pipes and finally ends up spilling on one side.

If the water spills on Alice's side we define the output to be 0. Otherwise, the water spills on Bob's side and the output value is 1. It is easy to see that due to the way the connections are made the water must eventually spill on one of the two sides, since cycles are not possible.

Note that the pipes can be viewed as a communication channel that can transmit $\log s$ bits, and that the garden-hose protocol is memoryless, i.e., regardless of the previous history, water from pipe i always flows to pipe j if those two pipes are connected. Furthermore computation is reversible, i.e., one can follow the path taken by the water backwards (e.g. by sucking the water back).

Buhrman et al. [BFSS13] have shown that it is possible to compute every function $f : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ by playing a garden-hose game. A garden-hose protocol consists of the scheme by which Alice chooses her connections depending on her private input $x \in \{0,1\}^n$ and how Bob chooses his connections depending on his private input $y \in \{0,1\}^n$. Alice also chooses the pipe that is connected to the tap. The protocol computes a function f, if for all inputs with $f(x,y) = 0$

the water spills on Alice's side, and for all inputs with $f(x, y) = 1$ the water spills on Bob's side.

The size of a garden-hose protocol is the number s of pipes used. The garden-hose complexity GH(f) of a function $f(x, y)$ is the minimum number of pipes needed in any garden-hose game that computes the value of f for all x and y such that $f(x, y)$ is defined.

The garden-hose model is originally motivated by an application to quantum position-verification schemes [BFSS13]. In this setting the position of a prover is verified via communications between the prover and several verifiers. An attack on such a scheme is performed by several provers, none of which are in the claimed position. Buhrman et al. [BFSS13] proposed a protocol for position-verification that depends on a function $f : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$, and a certain attack on this scheme requires the attackers to share as many entangled qubits as the garden-hose complexity of f. Hence all f with low garden-hose complexity are not suitable for this task, and it becomes desirable to find explicit functions with large garden-hose complexity.

Apart from the applications to position based quantum cryptography, the garden-hose model is quite appealing in itself. Garden-hose complexity can be viewed as a natural measure of space, in a situation where two players with private inputs compute a Boolean function cooperatively. And as we shall see in this chapter it has interesting connections to several other complexity measures and studying garden-hose model can lead to new lower bounds.

8.1.2 Formal Definition of the Model

We now describe the garden-hose model in graph terminology. In a garden-hose protocol with s pipes there is a set V of s vertices plus one extra vertex, the *tap* t.

Given their inputs x, y Alice and Bob want to compute $f(x, y)$. Depending on x Alice connects some of the vertices in $V \cup \{t\}$ in pairs by adding edges $E_A(x)$ that form a matching among the vertices in $V \cup \{t\}$. Similarly, Bob connects some of the vertices in V in pairs by adding edges $E_B(y)$ that form a matching in V.

Notice that after they have added the additional edges, a path starting from vertex t is formed in the graph $G = (V \cup \{t\}, E_A(x) \cup E_B(y))$. Since no vertex has degree larger than 2, this path is unique and ends at some vertex. We define the output of the game to be the parity of the length of the path starting at t. For instance, if the tap is not connected the path has length 0, and the output is 0. If the tap is connected to another vertex, and that vertex is the end of the path, then the path has length 1 and the output is 1 etc.

A garden-hose protocol for $f : \mathcal{X} \times \mathcal{Y} \to \{0,1\}$ is a mapping from $x \in \mathcal{X}$ to matchings among

$V \cup \{t\}$ together with a mapping from $y \in \mathcal{Y}$ to matchings among V. The protocol computes $f(x, y)$ if for all x, y the path has even length iff $f(x, y) = 0$. The garden-hose complexity of f is the smallest s such that a garden-hose protocol of size s exists that computes f.

We note that one can form a matrix G_s that has rows labelled by all of Alice's matchings, and columns labelled by Bob's matchings, and contains the parity of the path lengths. A function f has garden-hose complexity s iff its communication matrix is a sub-matrix of G_s. G_s is called the *garden-hose matrix* for size s.

8.1.3 Previous Work

Buhrman et al. [BFSS13] prove a number of results about the garden-hose model:

- Deterministic one-way communication complexity can be used to show lower bounds of up to $\Omega(n/\log n)$ for many functions.
- For the Equality problem they refer to a bound of $GH(Equality) = \Theta(n)$ shown by Pietrzak (the proof implicitly uses the fooling set technique from communication complexity [KN97] [personal communication]).
- They argue that super-polynomial lower bounds for the garden-hose complexity of a function f imply that the function cannot be computed in Logspace, making such bounds hard to prove for 'explicit' functions.
- They define randomized and quantum variants of the model and show that randomness can be removed at the expense of multiplying size by a factor of $O(n)$ (for quantum larger gaps are known).
- Via a counting argument it is easy to see that most Boolean functions need size $GH(f) = 2^{\Omega(n)}$.

Chiu et al. [CSWX13] improved the upper bound for the Equality function to $1.359n$ from the previously known $2n$ bound [BFSS13].

8.1.4 Garden-Hose Model and Communication Complexity

In this chapter we study garden-hose complexity and establish several new connections with well studied models like communication complexity, permutation branching programs, and formula size.

But before we begin we review some well studied complexity models. First we define Boolean formulae.

Definition 8.1.1. *A Boolean formula is a Boolean circuit whose every node has fan-out 1 (except the output gate). A Boolean formula of depth d is then a tree of depth d. The nodes are labelled by gate functions from a family of allowed gate functions, e.g. the class of the 16 possible functions of the form $f : \{0,1\} \times \{0,1\} \to \{0,1\}$ in case the fan-in is restricted to 2. Another interesting class of gate functions is the class of all symmetric functions (of arbitrary fan-in). The* formula size *of a function f (relative to a class of gate functions) is the smallest number of leaves in a formula computing f.*

Now we define branching programs. Our definition of permutation branching programs is extended in a slightly non-standard way.

Definition 8.1.2. *A branching program is a directed acyclic graph with one source node and two sink nodes (labelled with* `accept` *and* `reject`*). The source node has in-degree 0. The sink nodes have out-degree 0. All non-sink nodes are labelled by variables $x_i \in \{x_1, \cdots, x_n\}$ and have out-degree 2. The computation on an input x starts from the source node and depending on the value of x_i on a node either moves along the left outgoing edge or the right outgoing edge of that node. An input $x \in \{0,1\}^n$ is accepted iff the path defined by x in the branching program leads to the sink node labelled by* `accept`*. The length of the branching program is the maximum length of any path, and the size is the number of nodes.*

A layered branching program of length l is a branching program where all non-sink nodes (except the source) are partitioned into l layers. All the nodes in the same layer query the same variable x_i, and all outgoing edges of the nodes in a layer go to the nodes in the next layer or directly to a sink. The width of a layered branching program is defined to be the maximum number of nodes in any layer of the program. We consider the starting node to be in layer 0 and the sink nodes to be in layer l.

A permutation branching program is a layered branching program, where each layer has the same number k of nodes, and if x_i is queried in layer i, then the edges labelled with 0 between layers i and $i+1$ form an injective mapping from $\{1, \ldots, k\}$ to $\{1, \ldots, k\} \cup \{\texttt{accept}, \texttt{reject}\}$ (and so do the the edges labelled with 1). Thus, for permutation branching programs if we fix the value of x_i, each node on level $i+1$ has in-degree at most 1.

We call a permutation branching program strict *if there are no edges to* `accept`/`reject` *from*

internal layers. This is the original definition of permutation branching programs. Programs that are not strict are also referred to as loose *for emphasis.*

We denote by $PBP(f)$ the minimal size of a permutation branching program that computes f.

We note that simple functions like AND, OR can easily be computed by linear size loose permutation branching programs of width 2, something that is not possible for strict permutation branching programs [Bar85].

Lower Bound via Non-deterministic Communication

We start by showing that non-deterministic communication complexity gives lower bounds on the garden-hose complexity of any function f. This bound is often better than the bound $GH(f) \geq \Omega(D_1(f)/\log(D_1(f)))$ shown in [BFSS13], which cannot be larger than $n/\log n$. Thus for several important functions like Inner Product, Disjointness a lower bound of $\Omega(n)$ is achieved.

Theorem 8.1.3. $GH(f) \geq N(f) - 1$.

The main idea is that a nondeterministic protocol that simulates the garden-hose game can choose the *set* of pipes that are used on a path used on inputs x, y instead of the path itself, reducing the complexity of the protocol. The set that is guessed may be a superset of the actually used pipes, introducing ambiguity. Nevertheless we can make sure that the additionally guessed pipes form cycles and are thus irrelevant.

Proof. Consider a deterministic garden-hose protocol P for f using s pipes. Maybe the most natural approach to simulate P's computation by a non-deterministic communication protocol would be to guess the path that the water takes, and verify this guess locally by Alice and Bob. There are, however, too many paths for this to lead to good bounds. Instead we use a coarser guess. For any given input x, y in a computation of P the water traverses a set $W(x, y)$ of pipes. We refer to these pipes as the *wet* pipes in P on x, y. In general a set of wet pipes can correspond to several paths through the network, which must use only edges from the set.

In the non-deterministic protocol Alice guesses a set S of pipes that is supposed to be $W(x, y)$. Since $|W(x, u)|$ is odd if and only if $f(x, y) = 1$ the size of S immediately tells us whether S is a witness for 1-inputs or 0-inputs.

Consider an even size set S. Alice computes the connections of the pipes on her side using her input x (as used in the garden-hose protocol). Her connections are *consistent* with S, iff the tap is

connected to a pipe in S, and the other pipes in S are all connected in pairs, except one, which is open. Note that none of the pipes in S may be connected to a pipe outside of S. Similarly, S is consistent with Bob's connections (based on y), if all the pipes in S are paired up (no pipe in S is open and no pipe in S is connected to a pipe outside S).

For odd size S we use an analogous definition of consistency: Now Alice has no open pipe in S and all pipes in S are paired up except the one connected to the tap, and Bob has all pipes in S paired up except one that is open.

Suppose that S is consistent with the connections defined by x, y. Denote by $P(x, y)$ the path the water takes in the garden-hose protocol. We claim that all the pipes in $P(x, y)$ are in S, and that the remaining pipes in S form cycles. If this is the case then the non-deterministic protocol is correct: Since cycles have even length, subtracting them does not change the fact that $|S|$ is even or odd, and hence the size of S and $P(x, y)$ have the same parity, i.e., a consistent S determines the function value correctly. Also note that the communication complexity of the non-deterministic protocol is at most $GH(f)$+1, since a subset of the pipes used can be communicated with s bits: Alice guesses an S that is consistent with her input and sends it to Bob, who accepts/rejects if S is also consistent with his input, otherwise he gives up (accepting/rejecting takes one additional bit of communication). Note that for partial functions no consistent S may exist for Alice to choose, but in that case she can give up without a result.

To establish correctness we have to show that all pipes in $P(x, y)$ are in S (and the remaining pipes in S form cycles). Clearly the starting pipe (the one connected to the tap) is in S by the definition of consistency. All remaining pipes in S on Bob's and Alice's side are either paired up or (for exactly one pipe) open. Hence we can follow the flow of water without leaving S. This implies that $P(x, y)$ is in S, and since removing $P(x, y)$ from S leaves no open pipes all the remaining pipes in S must form a set of cycles. □

As an application consider the function $IP(x, y) = \sum_{i=1}^{n}(x_i \cdot y_i) \ mod\ 2$. It is well known that $N(IP) \geq n + 1$ [KN97], hence we get that $GH(IP) \geq n$. The same bound holds for Disjointness. These bounds improve on the previous $\Omega(n/\log n)$ bounds for these functions [BFSS13]. Furthermore note that the fooling set technique gives only bounds of size $O(\log^2 n)$ for the complexity of IP (see [KN97]), so the technique previously used to get a linear lower bound for Equality fails for IP.

$GH(f)$ At Most The Size of a Protocol Tree for f

Buhrman et al. [BFSS13] previously show that any one way communication complexity protocol with complexity $D_1(f)$ can be converted to a garden-hose protocol with $2^{D_1(f)}+1$ pipes. One-way communication complexity can be much larger than two-way communication [PS84]. It turns out that any 2-way deterministic communication protocol can also be converted to a garden-hose protocol so that the complexity $GH(f)$ is upper bounded by the *size* of the protocol tree of the communication protocol.

Theorem 8.1.4. *For any function f, the garden-hose complexity $GH(f)$ is upper bounded by the number of edges in a protocol tree for f.*

Proof. Given a protocol tree (with k edges) of a two way communication protocol P for any function f we construct a garden-hose protocol with at most k pipes.

We describe the construction in a recursive way. Let v be any node of the protocol tree belonging to Alice, with children $u_1, \ldots, u_d$ belonging to Bob. In the protocol tree rooted at v a function f_v is computed. If none of the u_i are leaves, then we assume by induction that we can construct a garden hose protocol P_i for each of the children, where P_i uses at most s_i many pipes, and s_i is the number of edges in the subtree of u_i. The P_i have the tap on Bob's side. To find a garden-hose protocol for v, we use $d + \sum s_i$ pipes. Alice sends the water through pipe i to communicate the message corresponding to the edge to u_i. Furthermore the right end of pipe i is connected to the tap of a copy of P_i. The number of pipes used $(d + \sum s_i)$ is at most the number of edges in the protocol tree. If one or two of the u_i are leaves, we use the same construction, except that for an accepting leaf we use one extra pipe that is open on Bob's end, and for a rejecting leaf we just let the water spill at Alice's pipe. It is easy to see by induction that the garden-hose protocol accepts on x, y if and only if the protocol tree ends in an accepting leaf. □

8.2 Garden-Hose Model and Other Complexity Classes

We now turn to comparing the garden-hose model to another nonuniform notion of space complexity, namely branching programs. In Section 8.2.1 we show how to convert any *permutation branching program* to a garden-hose protocol with only a constant factor loss in size.

The most important application of this simulation is that it allows us to find a garden-hose

protocol for the distributed Majority function,

$$DMAJ(x,y) = 1 \text{ iff } \sum_{i=1}^{n}(x_i \cdot y_i) \geq \frac{n}{2},$$

that has size $O(n \cdot \log^3 n)$, disproving the conjecture in [BFSS13] that this function has complexity $\Omega(n^2)$ (see Section 8.2.2).

Using the garden-hose protocols for Majority, Parity, AND, OR, we show upper bounds on the composition of functions with these.

Then in Section 8.2.3 we show how to convert any Boolean formula with AND, OR, XOR gates to a garden-hose protocol with a small loss in size. In particular, any formula consisting of arbitrary fan-in 2 gates only can be simulated by a garden-hose protocol with a constant factor loss in size. This result strengthens the previous observation that explicit super-polynomial lower bounds for $GH(f)$ will be hard to show: even bounds of $\Omega(n^{2+\epsilon})$ would improve on the long-standing best lower bounds on formula size due to Nečiporuk from 1966 [Nec66]. We can also simulate formulae including a limited number of Majority gates of arbitrary fan-in, so one might be worried that even super-linear lower bounds could be difficult to prove. We argue, however, that for formulae using arbitrary symmetric gates we can still get near-quadratic lower bounds using a Nečiporuk-type method. Nevertheless we have to leave super-linear lower bounds on the garden-hose complexity as an open problem.

8.2.1 Relating Permutation Branching Programs and the Garden-Hose Model

Before we begin we need to following notion of spilling pipes.

Definition 8.2.1. *In a garden-hose protocol a spilling-pipe on a player's side is a pipe such that water spills out of that pipe on the player's side during the computation for some input x, y.*

We say a protocol has multiple spilling-pipes if there is more than one spilling-pipe on Alice's side or on Bob's side.

We now show a technical lemma that helps us compose garden-hose protocols without blowing up the size too much.

Lemma 8.2.2. *A garden-hose protocol P for f with multiple spilling pipes can be converted to another garden-hose protocol P' for f that has only one spilling pipe on Alice's side and one spilling pipe on Bob's side. The size of P' is at most 3 times the size of P plus 1.*

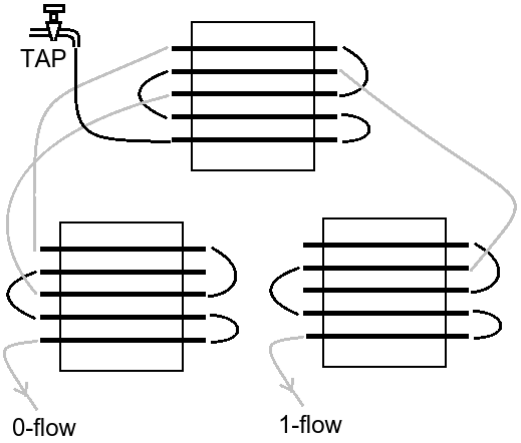


Figure 8.1: The Construction in Lemma 8.2.2

Proof. Fix a protocol P that uses s pipes to compute f. In the protocol P Alice makes the connections on her side based on her input x. Similarly, Bob's connections are based on his input y. Denote the set of pipes that are open on Alice's side by S_A and the set of pipes that are open on Bob's side by S_B.

In the new protocol P' Alice and Bob have $3s$ pipes arranged into 3 blocks of s pipes each. Let's call them B_1, B_2 and B_3. The main idea is to use B_1 to compute f and then use B_2 and B_3 to 'un-compute' f (to remove the extra information provided by the multiple spilling pipes).

In the construction of P' Alice and Bob make their connections on B_1, B_2 and B_3 separately, exactly the same way they did in P for s pipes. Alice then connects B_1's tap-pipe to the tap and keeps the tap-pipes of B_2, B_3 open. They then add the following connections: Alice connects every pipe $i \in S_A$ in B_1 to pipe $i \in S_A$ in B_2 and Bob connects every pipe $i \in S_B$ in B_1 to pipe $i \in S_B$ in B_3. Note that those pipes were open before they were connected as they were all spilling pipes. B_1 now does not have any open pipes. The only pipes that will ever spill in B_2 and B_3 are their taps (there may be other open pipes but it is easy to see that they never spill). The tap-pipes of B_2 and B_3 are both on Alice's side. Finally, Alice uses one more pipe, and connects the tap-pipe of B_3 to the new pipe. Figure 1 shows an example of the construction.

The size of the new protocol P' is exactly $3s + 1$, and there is exactly one spilling pipe on each side, namely the tap pipes of B_2 and B_3, because the only other open pipes are the S_A pipes in B_3 and the S_B pipes in B_2. These cannot be reached by the water. All connections made are done by Alice and Bob alone. We now argue that the protocol computes $f(x, y)$ correctly.

Notice that if $f(x, y) = 0$, then water flows through B_1 and ends at one of the pipes in S_A. This pipe is connected to the corresponding pipe in B_2. So the water follows the same path backwards

in B_2 until it reaches the tap-pipe in B_2. This pipe is open on Alice's side. Hence water spills on Alice's side making the output 0 (and it spills at the tap of B_2).

Similarly, if $f(x, y) = 1$, water flows through B_1 and ends at one of the pipes in S_B on Bob's side. Since this pipe is connected to the corresponding pipe in B_3 the water flows backwards din B_3 until it reaches the tap-pipe of B_3. This is on Alice's side and connected to the extra pipe. This makes the water to spill on Bob's side as desired. □

Now we are going to show that it is possible to convert a (loose) permutation branching program into a garden-hose protocol with only a constant factor increase in size. We are stating a more general fact, namely that the inputs to the branching program we simulate can be functions (with small garden-hose complexity) instead of just variables. This allows us to use composition.

Lemma 8.2.3. *$GH(g(f_1, f_2, ..., f_k)) = O(S \cdot \max(C_i)) + O(1)$, where $PBP(g) = S$ and $GH(f_i) = C_i$ and $f_i : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$. The f_i do not necessarily have the same inputs x, y.*

Proof. In Lemma 8.2.2 we have seen that we can turn a garden-hose protocol with multiple spilling pipes into a protocol with exactly one spilling pipe per side. Such a protocol acts exactly as a node in a branching program, except that its decision is based on $f_i(x_i, y_i)$. This observation suffices to simulate decision trees, but in a branching program nodes can have in-degree larger than 1, and we cannot pump water from several sources into a single garden-hose protocol.

We now show how to construct a garden-hose protocol for $g(f_1, f_2, ..., f_k)$. Given a loose permutation branching program for g of size S, we show how to construct a garden-hose protocol.

Let G denote the graph of the branching program. G consists of T layers $L_0, \ldots, L_{T-1}$, where the first layer has just one node (the source), the last layer 2 nodes (the sinks), and all intermediate layers have W nodes, so the size is $S = (T-2)W + 3$. Layer L_i queries some variable z_i, whose value is $f_i(x_i, y_i)$. The 1-edges between L_i and L_{i+1} are E_i^1, the 0-edges E_i^0.

The construction goes by replacing the nodes of each layer by the garden-hose protocols P_i for f_i. Each layer uses $2W$ copies of P_i, arranged in two layers. We refer to these copies as the upper and lower copies of P_i, each numbered from 1 to W (and implicitly by their level). Essentially we need the first layer to compute f_i, and the second layer to un-compute, since we only want to remember the name of the current vertex in G, not the value of f_i.

If $e = (j, k) \in E_i^1$, then we connect the 1-spill pipe of the upper j-th copy of P_i to the 1-spill pipe of the lower k-th copy of P_i. Similarly we make the connections for the 0-spill pipes (on

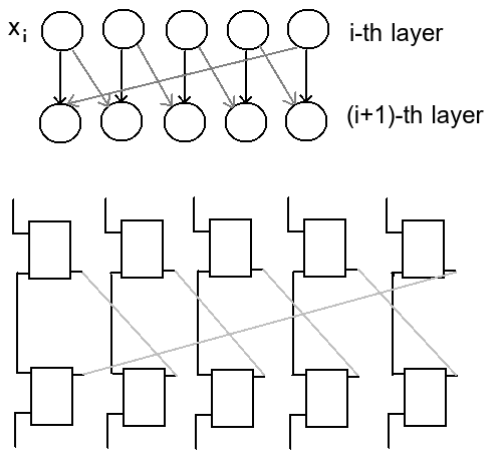


Figure 8.2: Permutation Branching Program to Garden-Hose Protocol Construction

Alice's side).

To connect layers we connect the tap-pipes on each lower copy j of a level L_i to the tap-pipes of an upper copy j on level L_{i+1}. On level L_0 the tap-pipe of an upper copy is connected to Alice's tap according to the branching program.

Figure 8.2 shows an example of the construction, where each block is a garden-hose protocol to compute f_i.

For every edge that goes to the accepting sink of the branching program we use one pipe that is connected to the corresponding upper copy on Alice's side, if the corresponding spilling pipe is on Alice's side. Otherwise we leave the spilling pipe open. We proceed analogously for edges to the rejecting sink.

The size of the garden-hose protocol is at most $2W \cdot \sum_{i=1}^{L} C_i \leq \max C_i \cdot 2WL$. □

A first corollary is the following fact already shown in [BFSS13]. Nonuniform Logspace is equal to the class of all languages recognizable by polynomial size families of branching programs. Since reversible Logspace equals deterministic Logspace [LMT00], and a reversible Logspace machine (on a fixed input length) can be transformed into a polynomial size permutation branching program, we get the following.

Corollary 8.2.4. *Logspace* $\subseteq GH(poly(n))$. *This holds for any partition of the variables among Alice and Bob.*

8.2.2 The Distributed Majority Function

In this section we investigate the complexity of the Distributed Majority function.

Definition 8.2.5. *Distributed Majority:* $DMAJ(x, y) = 1$ *iff* $\sum_i^n (x_i \cdot y_i) \geq \frac{n}{2}$, *where* $x, y \in \{0,1\}^n$.

Buhrman et al. [BFSS13] have conjectured that the complexity of this function is quadratic, which is what is suggested by the naïve garden-hose protocol for the problem. The naïve protocol implicitly keeps one counter for i and one for the sum, leading to quadratic size. Here we describe a construction of a permutation branching program of size $O(n \cdot \log^3 n)$ for Majority, which can then be used to construct a garden-hose protocol for the Distributed Majority function. The Majority function is defined by $MAJ(x_1, \ldots, x_1) = 1 \Leftrightarrow \sum x_i \geq n/2$.

Note that the Majority function itself can be computed in the garden-hose model using $O(n)$ pipes (for any way to distribute inputs to Alice and Bob), since Alice can just communicate $\sum_i x_i$ to Bob. The advantage of using a permutation branching program to compute Majority is that by Lemma 8.2.3 we can then find a garden-hose protocol for the composition of MAJ and the Boolean AND, which is the Distributed Majority function. We adapt a construction of Sinha and Thathachar [ST97], who describe a branching program for the Majority function.

Lemma 8.2.6. $PBP(MAJ) = O(n \cdot \log^3 n)$.

Proof. In 1997, Sinha et al. [ST97] described a branching program of size $O(\frac{n \log^3 n}{\log \log n \log \log \log n})$ for computing Majority. Unfortunately the branching program they construct is not a permutation branching program. Thus it is not immediately clear how to convert their construction into a garden-hose protocol.

To describe a permutation branching program for Majority we first need permutation branching programs for computing the sum of the inputs mod r for small r. Denote by Mod_r the (non-Boolean) function $Mod_r(x_1, \ldots, x_n) = \sum_i x_i \bmod r$. The following is easy to see.

Claim 8.2.7. $Mod_r(x_1, \ldots, x_n)$ *can be computed by permutation branching program of width* r *so that each input* x *with* $|x| = i$, *when starting on the top level at node* j *ends at node* $i + j \bmod r$ *on the last level.*

We call this permutation branching program a modulus-r box. The *join* of two modulus r_1 resp. r_2 boxes is a new branching program, in which bottom level nodes of the first box are identified in some way with top level nodes of the second. We employ the following main technical result of Sinha et al. [ST97], which describes an approximate divider.

Fact 8.2.8. *[ST97] Fix the length* M *of an interval of natural numbers. There are* $k \leq \log M$ *prime numbers* $r_2 < r_3 < \cdots < r_k$, *where* $r_2 > 4 \log M$ *and* $r_k < 12 \log M$ *and a number* $r_1 = 2^t$ *such that*

$2M \leq \Pi_{1 \leq i \leq k} r_i$ *and* $r_1 < r_2$. *Set* $M' = M/r_2$. *Consider inputs* x *such that* $b \leq |x| \leq b + M - 1$.

Then there is a way to join k *modulus-*r_i *boxes (in order* $r_1 \ldots, r_k$*) into a single branching program, such that all inputs* x *reaching the sink nodes named* lM' *mod* r_k *for some* $0 \leq l < r_k$ *(in the last box) satisfy that* $|x|$ *belongs to one of* r_2 *intervals of length* $(k-1)M'$ *in* $[b, b + M - 1]$. *The intervals overlap, and each point in* $[b, b + M - 1]$ *is in* $k - 1$ *intervals.*

Furthermore, the connections between the boxes are such that every output node of the r_i *box is connected to one input node of the* r_{i+1} *box, and every input node of the* r_{i+1} *box is connected to at most one output node of the* r_i *box.*

The above differs from the presentation in [ST97] in that we require that the r_i are increasing so that we can join them without creating nodes with fan-in larger than 1. This means that every r_i box for $i > 1$ has a few input nodes that are not used.

Note that our goal is to know whether $|x|$ is greater than $n/2$ or not. Effectively this means there are three kinds of bottom layer nodes in the branching program constructed above (for $b = 0$): those where we know that all inputs reaching the sink have $n/2 > |x|$, at which point we can reject, those where $n/2 < |x|$, where we accept, and undecided nodes. A bottom layer node is undecided, if the interval of possible $|x|$ reaching that sink contains $n/2$. At undecided nodes the interval of possible values of $|x|$ has been reduced to size $(k-1)M/r_2$, i.e., a $(k-1)/r_2 < 1/4$ fraction of the the original interval. Furthermore, there are $k - 1$ undecided nodes (since $n/2$ is in that many intervals), but the intervals for those nodes stretch to at most $(k-1)M/r_2$ beyond $n/2$ on both sides, hence the union of the intervals of all undecided bottom layer nodes is an interval of size at most $2(k-1)M/r_2 \leq M/2$. Hence, this construction can be iterated (at most $\log n$ times) to decide Majority on all inputs.

Now we need to argue that the whole construction can be made into a permutation branching program. Obviously any mod-r box can be computed by a strict permutation BP of width r and length n. The connections between the k boxed are injective mappings. Hence the whole constructions for the above fact can be made into a permutation branching program, where dummy nodes need to be added to bring all layers to the same width (r_k).

The branching program for Majority is then an iteration of the above construction of permutation branching programs. In each level of the iteration some nodes accept, some reject, and some continue on a smaller interval. For all undecided sink nodes we can assume that they continue using the same interval of size at most $M/2$. This continues until the intervals are very short ($M \leq \log n$), at which the problem can be solved by counting.

To do the same iteration in a permutation branching program we need to do the following. We want to turn a building block B_i of the iteration (a permutation branching program as in Fact 8.2.8) into a permutation branching program that has only 3 sinks reached by inputs (plus some sinks that are never reached). To do this we first use the original program, followed by 3 copies of the same program in reverse. We connect the undecided sinks of the upper program into the corresponding vertices in the first reversed lower program, similarly the accepting and rejecting sinks into the corresponding vertices of the other two reversed programs. Then each input that is undecided by B_i will end up at the node corresponding to the starting node of the first reverse copy. Similarly, inputs that are accepted by B_i will leave the second reverse copy at the node corresponding to the starting node of B_i etc. Using dummy nodes this program can be extended to a permutation branching program, with width increased by a factor of 3 and length by 2. Each input leads to one of three nodes. We can now connect the undecided sink of the above construction to the starting vertex of the next block B_{i+1}. To turn the whole construction into a strict permutation branching program the accepting and rejecting bottom vertices are connected to $O(n \log^2 n)$ extra vertices that remember at which layer/vertex the inputs were accepted/rejected.

The whole construction yields a permutation branching program for Majority. The length of the program is $O(\log n \cdot \log n \cdot n)$, for the $\log n$ iterations, the $k \leq \log n$ boxes that have length n. Each level of the program has width at most $O(\log M)$ for the mod r_i boxes and the constant factors to turn things into a permutation BP (plus $2 \log n$ vertices for accepting/rejecting paths)). Hence the total size of the program is $O(n \log^3 n)$.

□

We can now state the result about the composition of functions $f_1, \ldots, f_k$ with small garden-hose complexity via a Majority function.

Lemma 8.2.9. *For $(f_1, f_2, .., f_k)$, where each function f_i has garden-hose complexity $GH(f_i)$, we have $GH(MAJ(f_1, \ldots, f_k)) = O(\sum GH(f_i)) \cdot \log^3 k)$.*

The lemma immediately follows from combining Lemma 8.2.6 with Lemma 8.2.3. Considering $f_i = x_i \wedge y_i$ we get

Corollary 8.2.10. *The garden-hose complexity of distributed Majority is $O(n \log^3 n)$.*

8.2.3 Composition and Connection to Formula Size

We wish to relate $GH(f)$ to the formula size of f. To do so we examine composition of garden-hose protocols by popular gate functions.

Theorem 8.2.11. *For $(f_1, f_2, .., f_k)$, where each function f_i has garden-hose complexity $GH(f_i)$*

- $GH(\bigvee f_i) = O(\sum GH(f_i))$.
- $GH(\bigwedge f_i) = O(\sum GH(f_i))$.
- $GH(\oplus f_i) = O(\sum GH(f_i))$.
- $GH(MAJ(f_i)) = O(\sum GH(f_i) \cdot \log^3 k)$.

This result follows from Lemma 8.2.9 and Lemma 8.2.3 combined with the trivial loose permutation branching programs for AND, OR, XOR.

We now turn to the simulation of Boolean formulae by garden-hose protocols. We use the simulation of formulae over the set of all fan-in 2 functions by branching programs due to Giel [Gie01].

Theorem 8.2.12. *Let F be a formula for a Boolean function g on k inputs made of gates $\{\wedge, \vee, \oplus\}$ of arbitrary fan-in. If F has size s and $GH(f_i) \leq c$ for all i, then for all constants $\epsilon > 0$ we have $GH(g(f_1, f_2, .., f_k)) \leq O(s^{1+\epsilon} \cdot c)$.*

Proof. Giel [Gie01] shows the following simulation result:

Fact 8.2.13. *Let $\epsilon > 0$ be any constant. Assume there is a formula with arbitrary fan-in 2 gates and size s for a Boolean function f. Then there is a layered branching program of size $O(s^{1+\epsilon})$ and width $O(1)$ that also computes f.*

By inspection of the proof it becomes clear that the constructed branching program is in fact a strict permutation branching program. The theorem follows by applying Lemma 8.2.3. □

Corollary 8.2.14. *When the f_i's are single variables $GH(g) \leq O(s^{1+\epsilon})$ for all constants $\epsilon > 0$. Thus any lower bound on the garden-hose complexity of a function g yields a slightly smaller lower bound on formula-size (all gates of fan-in 2 allowed).*

The best lower bound of $\Omega(n^2/\log n)$ known for the size of formulae over the basis of all fan-in 2 gate function is due to Nečiporuk [Nec66]. The Nečiporuk lower bound method (based on

counting subfunctions) can also be used to give the best general branching program lower bound of $\Omega(n^2/\log^2 n)$ (see [Weg87]).

Due to the above any lower bound larger than $\Omega(n^{2+\epsilon})$ for the garden-hose model would immediately give lower bounds of almost the same magnitude for formula size and permutation branching program size. Proving super-quadratic lower bounds in these models is a long-standing open problem.

Due to the fact that we have small permutation branching programs for Majority, we can even simulate a more general class of formulae involving a limited number of Majority gates.

Theorem 8.2.15. *Let F be a formula for a Boolean function g on n inputs made of gates $\{\wedge, \vee, \oplus\}$ of arbitrary fan-in. Additionally there may be at most $O(1)$ Majority gates on any path from the root to the leaves. If F has size s, then for all constants $\epsilon > 0$ we have $GH(g) \leq O(s^{1+\epsilon})$.*

Proof. Proceeding in reverse topological order we can replace all sub-formulae below a Majority gate by garden-hose protocols with Theorem 8.2.12, increasing the size of the sub-formula. Then we can apply Lemma 8.2.9 to replace the sub-formula including the Majority gate by a garden-hose protocol. If the size of the formula below the Majority gate is $\tilde{s}$, then the garden-hose size is $O(\tilde{s}^{1+\epsilon'})$, where the poly-logarithmic factor of Lemma 8.2.9 is hidden in the polynomial increase. Since every path from root to leaf has at most $c = O(1)$ Majority gates, and we may choose the ϵ' in Theorem 8.2.12 to be smaller than ϵ/c, we get the desired result. □

8.2.4 The Nečiporuk Bound with Arbitrary Symmetric Gates

Since garden-hose protocols can even simulate formulae containing some arbitrary fan-in Majority gates, the question arises whether one can hope for super-linear lower bounds at all. Maybe it is hard to show super-linear lower bounds for formulae having Majority gates? Note that very small formulae for the Majority function itself are not known (the currently best construction yields formulae of size $O(n^{3.03})$ [Ser14]), hence we cannot argue that Majority gates do not add power to the model. In this subsection we sketch the simple observation that the Nečiporuk method [Nec66] can be used to give good lower bounds for formulae made of *arbitrary symmetric gates of any fan-in*. Hence there is no obstacle to near-quadratic lower bounds from the formula size connection we have shown. We stress that nevertheless we do not have any super-linear lower bounds for the garden-hose model.

We employ the communication complexity notation for the Nečiporuk bound from [Kla07].

Theorem 8.2.16. *Let $f : \{0,1\}^n \to \{0,1\}$ be a Boolean function and $B_1, \ldots, B_k$ a partition of the input bits of f. Denote by $D_j(f)$ the deterministic one-way communication complexity of f, when Alice receives all inputs except those in B_j, and Bob the inputs in B_j. Then the size (number of leaves) of any formula consisting of arbitrary symmetric Boolean gates is at least $\sum D_j(f)/\log n$.*

Proof. Fix f and $B_1, \ldots, B_k$ and any formula F of size at most n^2 computing f consisting of symmetric gates only. Define F_j to be the subtree of F, whose leaves are the variables in B_j (and root is the output gate of the formula, and denote by L_j the number of leaves of F_j. Then the size of F is $\sum L_j$. We will show that $D_j(f) \leq O(L_j \cdot \log n)$.

Alice has all the variables except those in B_j, which go to Bob. Alice (and Bob) have to evaluate all the gates in F_j (this includes the root). They will evaluate the gates in (reverse) topological order. All the leaves are known to Bob. Denote by P the set of paths in F_j that start at a leaf or a gate of fan-in at least 2 inside F_j, and end at a gate of fan-in at least 2 inside F_j and have no such gates in between. Then $L_j \geq |P|$. Also denote by G the set of gates in F_j that have fan-in larger than 1 inside F_j, again $L_j \geq |G|$. We will show that the communication is at most $O(L_j \cdot \log n)$.

Bob goes over paths $p \in P$ and gates in $g \in G$ in reverse topological order (i.e., from the leaves up). Let $p = v_1, \ldots, v_t$ be the vertices of some p in reverse topological order (i.e., the root is last). Denote by f_p the gate at v_{t-1}, the last vertex that has fan-in 1 in p. Alice can tell Bob which function is computed at v_{t-1} in terms of the value already computed (by Bob) at v_1. This takes 2 bits. Hence the total communication to evaluate paths in P is $2|P|$. For each $g \in G$ there are at least 2 inputs in F_j that have already been computed by Bob. Since the gate at g is symmetric, it is sufficient for Alice to say how many of her inputs to g evaluate to 1, which takes at most $2 \log n$ bits unless the formula is larger than n^2. So the total communication is at most $O(|P| + |G| \log n)$, and $|G|, |P| \leq L_j$, unless F has size larger than n^2 already.

□

The theorem is as good as the usual Nečiporuk bound except for the log-factor, and can hence be used to show lower bounds of up to $\Omega(n^2/\log^2 n)$ on the formula size of explicit functions like IndirectStorageAccess [Weg87].

8.3 Time Bounded Garden-Hose Protocols

We now define the notion of *time* in garden-hose protocols and prove that for any function f, if we restrict the number of times water can flow through pipes to some value k, we have $GH_k(f) = \Omega(2^{D_k(f)/k})$, where GH_k denotes the time-bounded garden-hose complexity, and D_k the k-round deterministic communication complexity. This result leads to strong lower bounds for the time bounded complexity of e.g. Equality, and to a time-hierarchy based on the pointer jumping problem.

To do this let us define the notion of *wet pipes*.

Definition 8.3.1. *Given a garden-hose protocol P for computing function f, and an input x, y we refer to the pipes that carry water in P on x, y as the wet pipes. Let T_P denote the maximum number of wet pipes for any input (x, y) in P.*

The number of wet pipes on input x, y is equal to the length of the path the water takes and thus corresponds to the time the computation takes. Thus it makes sense to investigate protocols which have bounded time T_P. Furthermore, the question is whether it is possible to simultaneously optimize T_P and the number of pipes used.

Definition 8.3.2. *We define $GH_k(f)$ to be the complexity of an optimal garden-hose protocol P for computing f where for any input (x, y) we have that T_P is bounded by k.*

As an example consider the Equality function (test whether $x = y$). The straightforward protocol that compares bit after bit has cost $3n$ but needs time $2n$ in the worst case. On the other hand one can easily obtain a protocol with time 2, that has cost $O(2^n)$: use 2^n pipes to communicate x to Bob. We have the following general lower bound.

Theorem 8.3.3. *For all Boolean functions f we have $GH_k(f) = \Omega(2^{D_k(f)/k})$, where $D_k(f)$ is the deterministic communication complexity of f with at most k rounds (Alice starting).*

Proof. We rewrite the claim as $D_k(f) = O(k \cdot \log GH_k(f))$.

Let P' be the garden-hose protocol for f that achieves complexity $GH_k(f)$ for f. The deterministic k-round communication protocol for f simulates P' by simply following the flow of the water. In each round Alice or Bob (alternatingly) send the name of the pipe used at that time by P'. □

Thus for Equality we have for instance that $GH_{\sqrt{n}}(Equality) = \Omega(2^{\sqrt{n}})$. There is an almost matching upper bound of $GH_{\sqrt{n}}(Equality) = O(2^{\sqrt{n}} \cdot \sqrt{n})$ by using $\sqrt{n}$ blocks of $2^{\sqrt{n}}$ pipes to communicate blocks of $\sqrt{n}$ bits each.

We can easily deduce a time-cost tradeoff from the above: For Equality the product of time and cost is at least $\Omega(n^2/\log n)$, because for time $T < o(n/\log n)$ we get a super-linear bound on the size, whereas for larger T we can use that the size is always at least n.

8.3.1 A Time-Size Hierarchy

The Pointer Jumping Function is well-studied in communication complexity. We describe a slight restriction of the problem in which the inputs are permutations of $\{1, \ldots, n\}$.

Definition 8.3.4. *Let U and V be two disjoint sets of vertices such that $|U| = |V| = n$.*

Let $F_A = \{f_A | f_A : U \to V$ and f_A is bijective$\}$ and $F_B = \{f_B | f_B : V \to U$ and f_B is bijective$\}$. For a pair of functions $f_A \in F_A$ and $f_B \in F_B$ define

$$f(v) = \begin{cases} f_A(v) & \text{if } v \in U \\ f_B(v) & \text{if } v \in V. \end{cases}$$

Then $f_0(v) = v$ and $f_k(v) = f(f_{k-1}(v))$.

Finally, the pointer jumping function $PJ_k : F_A \times F_B \to \{0,1\}$ is defined to be the XOR of all bits in the binary name of $f_k(v_0)$, where v_0 is a fixed vertex in U.

Round-communication hierarchies for PJ_k or related functions are investigated in [NW93]. Here we observe that PJ_k gives a time-size hierarchy in the garden-hose model. For simplicity we only consider the case where Alice starts.

Theorem 8.3.5. *For the pointer jumping function $PJ_k : F_A \times F_B \to \{0,1\}$ we have:*

1. *PJ_k can be computed by a garden-hose protocol with time k and size kn.*
2. *Any garden-hose protocol for PJ_k that uses time at most $k-1$ has size $2^{\Omega(n/k)}$ for all $k \leq n/(100 \log n)$.*

Proof. To show part 1) we use a protocol using nk pipes, organized into k blocks. If Alice has input f_A, then she connects the tap to pipe $f_A(v_1)$ in block 1. For all even numbered blocks $2j$ she connects the ith pipe in block $2j$ to pipe $f_A(i)$ in block $2j+1$. Bob connects for all odd numbered blocks the ith pipe in block $2j+1$ to pipe $f_B(i)$ in block $2j+2$.

Assume that k is odd. Then the kth vertex of the path is on Bob's side. If $PJ_k(f_A, f_B) = 0$ then the XOR of $f_k(v_0)$ is 0 and the water needs to spill on Alice's side. Hence, in block k, for all pipes i with even i, Alice leaves the pipe open instead of connecting it to a pipe in block $k-1$. She does make the connections as described above for all odd pipes in block k.

Similarly, if k is even, then the last vertex is on Alice's side and if $f_k(v_0)$ is odd the spill needs to be on Bob's side. Hence Bob skips all the connections between blocks $k-1$ and k for odd numbered pipes i in block k.

Note that f_A and f_B are bijective, hence the connections made are legal. In total we use kn pipes. It is clear that the garden-hose protocol described above computes PJ_k.

Now we turn to part 2. Take any time $k-1$ garden-hose protocol for PJ_k using s pipes. Due to the simulation in Theorem 8.3.3 we get a $k-1$ round communication protocol (Alice starting) with communication $(k-1)\log s$. But Nisan and Wigderson [NW93] show that such protocols need communication $\Omega(n)$ for $k \leq n/(100 \log n)$. Hence $s \geq 2^{\Omega(n/k)}$.

The difficulty in applying their result is that Nisan and Wigderson analyse the complexity of PJ_k for uniformly random inputs, not random *bijective* inputs f_A resp. f_B. Hence we need to make some changes to their proof. These changes needed to make the argument work are minor, however: the uniform distribution on pairs of bijective functions is still a product distribution, and as long as $k = o(n)$ it is still true that at any vertex in the protocol tree the information about the next pointer is a small constant. The main difference to the original argument is that conditioning on the previous path introduces information about the next pointer due to the fact that vertices on the path can not be used again. This can easily be subsumed into the information given via the previous communication. □

We note that slightly weaker lower bounds hold for the randomized setting.

8.4 Randomized Garden-Hose Protocols

We now bring randomness into the picture and investigate its power in the garden-hose model. Buhrman et al [BFSS13] have already considered protocols with public randomness. In this section we are mainly interested in the power of private randomness.

Definition 8.4.1. *Let $RGH^{pub}(f)$ denote the minimum complexity of a garden-hose protocol for computing f, where the players have access to public randomness, and the output is correct*

with probability 2/3 (over the randomness). Similarly, we can define $RGH^{pri}(f)$, the cost of garden-hose protocols with access to private randomness.

By standard fingerprinting ideas [KN97] we can observe the following.

Claim 8.4.2. $RGH^{pub}(Equality) = O(1)$

Claim 8.4.3. $RGH^{pri}(Equality) = O(n)$*, and this is achieved by a constant time protocol.*

Proof. The second claim follows from Newman's theorem [New91] showing that any public coin protocol with communication cost c can be converted into a private coin protocol with communication cost $c + \log n + O(1)$ bits on inputs of length n together with the standard public coin protocol for Equality, and the protocol tree simulation of Theorem 8.1.4. □

Of course we already know that even the deterministic complexity of Equality is $O(n)$, hence the only thing achieved by the above protocol is the reduction in time complexity. Note that due to the result in the previous section computing Equality deterministically in constant time needs exponentially many pipes.

Buhrman et al. [BFSS13] have shown how to de-randomize a public coin protocol at the cost of increasing size by a factor of $O(n)$, so the factor n in the separation between public coin and deterministic protocols above is the best that can be achieved. This raises the question whether private coin protocols can ever be more efficient in size than the optimal deterministic protocol. We now show that there are no very efficient private coin protocols for Equality.

Claim 8.4.4. $RGH^{pri}(Equality) = \Omega(\sqrt{n}/\log n)$

Proof. To prove this we first note that $RGH^{pri}(f) = \Omega(R^{||}(f)/\log R^{||}(f))$, where $R^{||}(f)$ is the cost of randomized private coin simultaneous message protocols for f (Alice and Bob can send their connections to the referee). Hence, $RGH^{pri}(f) = \Omega(R^{||}(f)/\log R^{||}(f))$, but Newman and Szegedy [NS96] show that $RGH^{pri}(Equality) = \Omega(\sqrt{n})$. □

Chapter 9

Conclusion and Open Problems

9.1 Introduction

In this thesis, we have studied two different topics. The first topic is concerned with graph properties in the query complexity models. The second topic is concerned with communication complexity and garden-hose protocols. Here we briefly recall the main results of each chapter and list some related open problems for further study.

9.1.1 Query Complexity of Graph Properties

In Chapter 5 we've obtained an $\Omega(n^{3/4})$ lower bound for the quantum query complexity of Subgraph Isomorphism Problem for graphs, improving upon previously known $\Omega(n^{2/3})$ bound for the same. We have also extended our result to the 3-uniform hypergraphs by exhibiting an $\Omega(n^{4/5})$ bound, which improves a previously known $\Omega(n^{3/4})$ bound. Besides the obvious question of settling the randomized and quantum query complexity of the Subgraph Isomorphism problem, there are a few interesting questions that might be approachable.

Question 9.1.1. *Is it true that for any n-vertex graph H we have:*

(a) $R(f_H) = \Omega(\alpha_H \cdot n)$*?*

(b) $R(f_H) = \Omega(n^2/d^H_{avg}))$*?*

(c) $R(f_H) = \Omega(n^2/\chi_H)$*?*

Question 9.1.2. *Is it true that for any 3-uniform hypergraph H we have:*

$$Q(f_H) = \Omega(n)?$$

Note that, in the proof of Theorem 5.2.8 we've managed to get a slightly stronger $\Omega(n^{5/6})$ bound for the case 2. Thus an improved lower bound of $\Omega(n^{5/6})$ for the case 1 (when $\alpha_H > n/2$) would improve the overall bound.

9.1.2 Graph Properties in Node Query Settings

In Chapter 6 we have investigated query complexity of graph properties in the node query settings. It allowed us to initiate a systematic study of breaking symmetry. We have found that with a minimal symmetry assumption on G the query complexity for any hereditary graph property in that setting is the worst possible. We've also shown that in the absence of any symmetry on G every hereditary property benefits at least quadratically. Furthermore it cannot fall exponentially low for any hereditary property. We've also studied several other important graph properties in this setting. Below we list some of the open problems related to the node query setting.

- **Weak-evasiveness in the presence of symmetry:** Is it true that every hereditary graph property $\mathcal{P}$ in the node-query setting is weakly-evasive under symmetry, i.e., $\mathcal{G}_{\mathcal{T}}\text{-}min\text{-}cost(\mathcal{P}) = \Omega(n)$? What about the randomized case i.e., is it true that even in the randomized query complexity settings every hereditary graph property is weakly-evasive?

- **Polynomial lower bound in the absence of symmetry:** How low can $min\text{-}cost(\mathcal{P})$ fall for a hereditary $\mathcal{P}$ in the absence of symmetry? In this thesis we have proved a lower bound of $\Omega(\log n/\log\log n)$. Is it possible to improve the bound substantially?

- **Further restrictions on graphs:** While studying restricted graph classes we have given more stress in planar graphs. Now the question is how low can $\mathcal{G}\text{-}min\text{-}cost(\mathcal{P})$ fall for hereditary properties $\mathcal{P}$ on restricted classes of graphs $\mathcal{G}$ such as social-network graphs, planar graphs, bipartite graphs, bounded degree graphs etc?

- **Tight bounds on min-$cost$:** What are the tight bounds for natural properties such as acyclicity, planarity, containing a cycle of length t, path of length t?

- **Extension to hypergraphs:** We have studied graph properties in the node query framework. But we have completely untouched the hypergraph regime. It would be very interesting to investigate interesting properties for instance hereditary graph properties for hypergraphs. For instance, we can even focus on 3-uniform hypergraph and ask what happens for hereditary properties of (say) 3-uniform hypergraphs in node-query setting? We

note that $min\text{-}cost(\mathcal{I}) = \Theta(n^{1/3})$ for 3-uniform hypergraphs. What about other properties?

- **Global vs local:** Note that global connectivity requires $\Theta(n)$ queries whereas the cost of s-t connectivity for fixed s and t can be as low as $O(1)$. What about other properties such as min-cut?

- **How about max-$cost$ upper bounds? :** From algorithmic point of view, it might be interesting to obtain good upper bounds on the $max\text{-}cost(\mathcal{P})$ for some natural properties. It might also be interesting to investigate $\mathcal{G}\text{-}max\text{-}cost(\mathcal{P})$ for several restricted graph classes such as social-network graphs, planar graphs, bipartite graphs etc. In our opinion this would be very interesting topic for future investigations.

9.1.3 Power of Quantum Messages and Proofs in Communication Protocols

Chapter 7 has dealt with quantum messages and quantum proofs. We have investigated the relationship between quantum and classical messages and shown how to replace a quantum message in a one-way communication protocol by a deterministic message. Thus we've established that for all partial Boolean functions $f : \{0,1\}^n \times \{0,1\}^m \to \{0,1\}$ we have

$$D^{A\to B}(f) \leq O(Q^{A\to B,*}(f) \cdot m).$$

We have also shown that QMA in one-way communication complexity is exponentially more powerful than QCMA in communication complexity model by exhibiting a partial Boolean function f for which a QMA protocol has cost $O(\log n)$, whereas every QCMA protocol for f requires communication $\Omega(\sqrt{n}/\log n)$.

Below we discuss some of the very interesting open problems related to this topics:

- Aaronson [Aar07] argued that the bound in Theorem 7.1.3 of Chapter 7 is tight to within polylogarithmic factors for partial functions. However, a long-standing conjecture is that for all total Boolean functions f the following is true

$$R^{A\to B}(f) = O(Q^{A\to B}(f)),$$

or something weaker might also be true, i.e., for total Boolean functions we can remove the factor m completely, but possibly at the expense of increasing the dependence on $Q^{A\to B}(f)$

polynomially. An easier problem might be to replace m by something smaller like $\sqrt{m}$ for total functions.

- For many "nondeterministic" modes of communication complexity one-way communication is as good as two-way communication, for instance for nondeterministic, QMA, AM-complexity. We have proved that this is not the case for QCMA protocols (and this was known previously for MA-protocols [Kla11]). Is there a proper round-hierarchy for QCMA or MA protocols, i.e., is it true that there is a function that can be computed efficiently in k rounds but not in $k-1$ rounds?

- MA-communication complexity has recently been applied to the analysis of cloud-computing on data-streams and related topics [CCM+13, CCMT14]. Currently we don't have lower bounds larger than $\sqrt{n}$ for MA-communication complexity of explicit functions, while counting arguments show that most functions have complexity $\Omega(n)$. This gap is quite significant in practice. Can larger bounds be shown for an explicit function, at least in the one-way model, or the even more restricted *online* one-way model?

- Lower bounds for the AM-communication complexity of any explicit function remain elusive.

- It would be interesting to separate QCMA- and QMA- communication complexity in the general two-way communication model. Such a separation could be used in the algebrization framework [AW09] to argue that showing QCMA=QMA (in the Turing machine world) would require nonalgebrizing techniques, and might also be thought of as evidence that these classes are not equal after all.

- A similar problem, and probably less ambitious, is to show that for a Boolean function f the QMA and QCMA are different in query complexity model. However, while for communication complexity we have a good candidate for such a separation (the QMA-complete problem), we are not aware of any good candidate for the query complexity setting. Is there a complete (promise) problem for QMA-query complexity?

9.1.4 Garden-hose Protocols

In Chapter 8 we have studied the garden-hose model and shown several new connections between this models and other well-established computational models. Some of the main results includes

having improved lower bounds based on the non-deterministic communication complexity, as well as a new $O(n \cdot \log^3 n)$ upper bound for the Distributed Majority function and an efficient simulation of formulae made of AND, OR, XOR gates in the garden-hose model. We have also studied a time-bounded variant of the garden-hose model and have shown how to simulate any permutation branching program by a garden-hose protocol. As this is a new computation model several interesting questions can be studied in this model. In particular we would like to highlight the following few questions:

- We have argued that getting lower bounds on $GH(f)$ larger than $\Omega(n^{2+\epsilon})$ will be difficult for any function f. But we know of no obstacles to proving super-linear lower bounds. Thus it would be very interesting to see if we can achieve super-linear lower bounds on the size of garden hose protocols.

  We think a possible candidates for quadratic lower bounds could be the Disjointness function with set size n and universe size n^2, and the IndirectStorageAccess function [Weg87].

- Consider the garden-hose matrix G_s as a communication matrix. How many distinct rows does G_s have? What is the deterministic communication complexity of G_s? The best upper bound is $O(s \log s)$, and the lower bound is $\Omega(s)$. An improved lower bound would give a problem, for which $D(f)$ is larger than $GH(f)$.

- We have proved $RGH^{pri}(Equality) = \Omega(\sqrt{n}/\log n)$. Is it true that $RGH^{pri}(Equality) = \Theta(n)$? Is there any problem where $RGH^{pri}(f)$ is smaller than $GH(f)$?

- The garden-hose model is a memoryless distributed model, but it is also reversible. Thus it would be interesting to investigate the relation between the garden-hose model and memoryless communication complexity, i.e., a variant of communication complexity model in which, unlike the standard two-way communication complexity (where each message is generated and sent based on all the previous transcripts and the private input), Alice and Bob must send messages depending only on their input and the message that they just received.

- The best (known) lower bound on the branching program size of an explicit Boolean function is $\Omega(n^2/\log^2 n)$, due to Nečiporuk [Nec66]. Of course, by using a counting argument it is easy to see that there exist functions requiring exponential size branching programs. Nevertheless the above bound has not been improved for half a century. Is it possible to

get a larger branching program size lower bound for some explicit function for *permutation* branching programs?

9.2 Concluding Remarks

In this thesis we have covered a few topics of two broad areas of complexity theory namely communication complexity and query complexity. We have also studied garden-hose complexity, a new promising model. These areas are rich with many fascinating and interesting open questions. Despite much effort a lot of these questions are still open and need further studies. Investigation of these questions would lead to new advancement of complexity theory and would broaden our understanding of time and space complexity and their trade-offs in general. In our opinion these areas of complexity theory would remain very interesting topics for future research.

Chapter 10

Bibliography

[Aar05] Scott Aaronson. Limitations of quantum advice and one-way communication. 1:1–28, 2005. Earlier version in Complexity'04. quant-ph/0402095. (page 9, 89, 90)

[Aar07] Scott Aaronson. The learnability of quantum states. In *Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, volume 463, pages 3089–3114. The Royal Society, 2007. (page 9, 89, 127)

[ABB+15] Andris Ambainis, Kaspars Balodis, Aleksandrs Belovs, Troy Lee, Miklos Santha, and Juris Smotrovs. Separations in query complexity based on pointer functions. *arXiv preprint arXiv:1506.04719*, 2015. (page 39, 40)

[ABDK15] Scott Aaronson, Shalev Ben-David, and Robin Kothari. Separations in query complexity using cheat sheets. *arXiv preprint arXiv:1511.01937*, 2015. (page 4, 39, 40, 53)

[AK07] Scott Aaronson and Greg Kuperberg. Quantum versus classical proofs and advice. *Theory of Computing*, 3(1):129–157, 2007. (page 10, 97)

[AKK16] Andris Ambainis, Martins Kokainis, and Robin Kothari. Nearly optimal separations between communication (or query) complexity and partitions. In *LIPIcs-Leibniz International Proceedings in Informatics*, volume 50. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2016. (page 45)

[AKP+82] Miklós Ajtai, János Komlós, Janos Pintz, Joel Spencer, and Endre Szemerédi. Extremal uncrowded hypergraphs. *Journal of Combinatorial Theory, Series A*, 32(3):321–335, 1982. (page 60, 61)

[Alo85] Noga Alon. Hypergraphs with high chromatic number. *Graphs and Combinatorics*, 1(1):387–389, 1985. (page 63)

[Amb03] Andris Ambainis. Polynomial degree vs. quantum query complexity. In *Foundations of Computer Science, 2003. Proceedings. 44th Annual IEEE Symposium on*, pages 230–239. IEEE, 2003. (page 5, 54)

[AN02] D. Aharonov and T. Naveh. "quantum np 96 a survey". quant-ph/0210077, 2002. (page 10, 97)

[ANTV99] A. Ambainis, A. Nayak, A. Ta-Shma, and U. Vazirani. Dense quantum coding and a lower bound for 1-way quantum automata. In *Proceedings of 31st ACM STOC*, pages 697–704, 1999. (page 88)

[AUY83] Alfred V Aho, Jeffrey D Ullman, and Mihalis Yannakakis. On notions of information transfer in vlsi circuits. In *Proceedings of the fifteenth annual ACM symposium on Theory of computing*, pages 133–139. ACM, 1983. (page 45)

[AW09] Scott Aaronson and Avi Wigderson. Algebrization: A new barrier in complexity theory. *ACM Transactions on Computation Theory (TOCT)*, 1(1):2, 2009. (page 128)

[Bab15] László Babai. Graph isomorphism in quasipolynomial time. *CoRR*, abs/1512.03547, 2015. (page XII, 52)

[Bar85] David A. Barrington. Width-3 permutation branching programs, 1985. Technical report, MIT/LCS/TM-293. (page 108)

[BBC+01] Robert Beals, Harry Buhrman, Richard Cleve, Michele Mosca, and Ronald de Wolf. Quantum lower bounds by polynomials. *Journal of the ACM (JACM)*, 48(4):778–797, 2001. (page 4, 5, 38, 53, 54)

[BCdWZ99] Harry Buhrman, Richard Cleve, Ronald de Wolf, and Christof Zalka. Bounds for small-error and zero-error quantum algorithms. In *Foundations of Computer Science, 1999. 40th Annual Symposium on*, pages 358–368. IEEE, 1999. (page 4, 5, 51)

[BCP+13] Joshua Brody, Shiteng Chen, Periklis A. Papakonstantinou, Hao Song, and Xiaoming Sun. Space-bounded communication complexity. In *Proceedings of the 4th conference on Innovations in Theoretical Computer Science*, pages 159–172, 2013. (page 11, 104)

[BCW98] Harry Buhrman, Richard Cleve, and Avi Wigderson. Quantum vs. classical communication and computation. In *Proceedings of the thirtieth annual ACM symposium on Theory of computing*, pages 63–68. ACM, 1998. (page 48)

[BDKP15] Nikhil Balaji, Samir Datta, Raghav Kulkarni, and Supartha Podder. Graph properties in node-query setting: effect of breaking symmetry. *arXiv preprint arXiv:1510.08267*, 2015. (page 14, 65)

[BdW02] Harry Buhrman and Ronald de Wolf. Complexity measures and decision tree complexity: a survey. *Theoretical Computer Science*, 288(1):21–43, 2002. (page 28, 30, 35, 36, 38)

[BFSS13] Harry Buhrman, Serge Fehr, Christian Schaffner, and Florian Speelman. The garden-hose model. In *Proceedings of the 4th conference on Innovations in Theoretical Computer Science*, pages 145–158. ACM, 2013. (page XIV, 11, 12, 13, 103, 104, 105, 106, 108, 109, 110, 111, 114, 115, 123, 124)

[BGK15] Ralph Bottesch, Dmitry Gavinsky, and Hartmut Klauck. Equality, revisited. In *International Symposium on Mathematical Foundations of Computer Science*, pages 127–138. Springer, 2015. (page 90)

[Bla15] Timothy Black. Monotone properties of k-uniform hypergraphs are weakly evasive. In *Proceedings of the 2015 Conference on Innovations in Theoretical Computer Science*, pages 383–391. ACM, 2015. (page 71)

[BTY94] Paul Beame, Martin Tompa, and Peiyuan Yan. Communication-space tradeoffs for unrestricted protocols. 23(3):652–661, 1994. Earlier version in FOCS'90. (page 11, 104)

[CCM+13] Amit Chakrabarti, Graham Cormode, Andrew McGregor, Justin Thaler, and Suresh Venkatasubramanian. On interactivity in arthur-merlin communication and stream computation. In *Electronic Colloquium on Computational Complexity (ECCC)*, volume 20, page 180, 2013. (page 128)

[CCMT14] Amit Chakrabarti, Graham Cormode, Andrew McGregor, and Justin Thaler. Annotations in data streams. *ACM Transactions on Algorithms (TALG)*, 11(1):7, 2014. (page 128)

[Cha05] Sourav Chakraborty. On the sensitivity of cyclically-invariant boolean functions. In *Computational Complexity, 2005. Proceedings. Twentieth Annual IEEE Conference on*, pages 163–167. IEEE, 2005. (page 71)

[CK01] Amit Chakrabarti and Subhash Khot. Improved lower bounds on the randomized complexity of graph properties. In *Automata, Languages and Programming*, pages 285–296. Springer, 2001. (page 3, 51)

[CKS01] Amit Chakrabarti, Subhash Khot, and Yaoyun Shi. Evasiveness of subgraph containment and related properties. *SIAM Journal on Computing*, 31(3):866–875, 2001. (page 70)

[CSWX13] Well Y Chiu, Mario Szegedy, Chengu Wang, and Yixin Xu. The garden hose complexity for the equality function. *arXiv:1312.7222*, 2013. (page 106)

[CT12] Thomas M Cover and Joy A Thomas. *Elements of information theory*. John Wiley & Sons, 2012. (page 25)

[Dat09] Nilanjana Datta. Min- and max- relative entropies and a new entanglement monotone. *IEEE Transactions on Information Theory*, 55:2816–2826, 2009. (page 26)

[Die92] Gröger Hans Dietmar. On the randomized complexity of monotone graph properties. *Acta Cybernetica*, 10(3):119–127, 1992. (page 4, 52, 53)

[DKSS13] Anindya De, Piyush P Kurur, Chandan Saha, and Ramprasad Saptharishi. Fast integer multiplication using modular arithmetic. *SIAM Journal on Computing*, 42(2):685–699, 2013. (page 1)

[dW02] Ronald de Wolf. Quantum communication and complexity. *Theoretical computer science*, 287(1):337–353, 2002. (page 49)

[ER60] Paul Erdös and Richard Rado. Intersection theorems for systems of sets. *Journal of the London Mathematical Society*, 1(1):85–90, 1960. (page 75)

[FKW02a] Ehud Friedgut, Jeff Kahn, and Avi Wigderson. Computing graph properties by randomized subcube partitions. In *Randomization and approximation techniques in computer science*, pages 105–113. Springer, 2002. (page 4, 51, 56, 57)

[FKW02b] Ehud Friedgut, Jeff Kahn, and Avi Wigderson. Computing graph properties by randomized subcube partitions. In *Randomization and approximation techniques in computer science*, pages 105–113. Springer, 2002. (page 7, 68)

[Gie01] Oliver Giel. Branching program size is almost linear in formula size. *Journal of Computer and System Sciences*, 63(2):222–235, 2001. (page 12, 118)

[GJPW15] Mika Göös, TS Jayram, Toniann Pitassi, and Thomas Watson. Randomized communication vs. partition number. In *Electronic Colloquium on Computational Complexity (ECCC) TR15*, volume 169, page 3, 2015. (page 39)

[GKK+08] Dmitry Gavinsky, Julia Kempe, Iordanis Kerenidis, Ran Raz, and Ronald de Wolf. Exponential separation for one-way quantum communication complexity, with applications to cryptography. *SIAM J. Comput.*, 38(5):1695–1708, 2008. (page 9, 88)

[GPW15] Mika Goos, Toniann Pitassi, and Thomas Watson. Deterministic communication vs. partition number. In *Foundations of Computer Science (FOCS), 2015 IEEE 56th Annual Symposium on*, pages 1077–1088. IEEE, 2015. (page 39)

[GR13] Chris Godsil and Gordon F Royle. *Algebraic graph theory*, volume 207. Springer Science & Business Media, 2013. (page 72)

[GRdW08] Dmitry Gavinsky, Oded Regev, and Ronald de Wolf. Simultaneous Communication Protocols with Quantum and Classical Messages. *Chicago Journal of Theoretical Computer Science*, 7, 2008. (page 90)

[GSS13] Justin Gilmer, Michael Saks, and Sudarshan Srinivasan. Composition limits and separating examples for some boolean function complexity measures. In *Computational Complexity (CCC), 2013 IEEE Conference on*, pages 185–196. IEEE, 2013. (page 34, 36)

[Haj91] Péter Hajnal. An (n 4/3) lower bound on the randomized complexity of graph properties. *Combinatorica*, 11(2):131–143, 1991. (page 3, 51)

[HJ85] Roger A Horn and Charles R Johnson. Matrix analysis cambridge university press. *New York*, 1985. (page 20)

[HOT81] Fumio Hiai, Masanori Ohya, and Makoto Tsukada. Sufficiency, kms condition and relative entropy in von neumann algebras. *Pacific Journal of Mathematics*, 96(1):99–109, 1981. (page 94)

[Iva] Gabör Ivanyos. *Personal communication.* (page 7, 68)

[JJUW11] Rahul Jain, Zhengfeng Ji, Sarvagya Upadhyay, and John Watrous. Qip = pspace. *J. ACM*, 58(6), 2011. (page 10, 96)

[JZ09] Rahul Jain and Shengyu Zhang. New bounds on classical and quantum one-way communication complexity. *Theoretical Computer Science*, 410(26):2463–2477, 2009. (page 88)

[Kin88] Valerie King. Lower bounds on the complexity of graph properties. In *Proceedings of the twentieth annual ACM symposium on Theory of computing*, pages 468–476. ACM, 1988. (page 3, 51)

[Kla00] Hartmut Klauck. On quantum and probabilistic communication: Las Vegas and one-way protocols. In *Proceedings of 32nd ACM STOC*, pages 644–651, 2000. (page 9, 88, 90)

[Kla04] Hartmut Klauck. Quantum and classical communication-space tradeoffs from rectangle bounds. In *Proceedings of FSTTCS*, 2004. (page 11, 104)

[Kla07] Hartmut Klauck. One-Way Communication Complexity and the Nečiporuk Lower Bound on Formula Size. *SIAM J. Comput.*, 37(2):552–583, 2007. (page 119)

[Kla11] Hartmut Klauck. On arthur merlin games in communiction complexity. In *IEEE Conference on Computational Complexity*, 2011. (page 98, 128)

[KN97] Eyal Kushilevitz and Noam Nisan. *Communication Complexity*. Cambridge University Press, 1997. (page 11, 46, 47, 49, 106, 109, 124)

[KNTSZ01] Hartmut Klauck, Ashwin Nayak, Amnon Ta-Shma, and David Zuckerman. Interaction in quantum communication complexity. In *Proceedings of 33rd ACM STOC*, 2001. (page 94)

[KP14a] Hartmut Klauck and Supartha Podder. New bounds for the garden-hose model. *arXiv preprint arXiv:1412.4904*, 2014. (page 13, 103)

[KP14b] Hartmut Klauck and Supartha Podder. Two results about quantum messages. In *International Symposium on Mathematical Foundations of Computer Science*, pages 445–456. Springer, 2014. (page 13, 87)

[KP16] Raghav Kulkarni and Supartha Podder. Quantum query complexity of subgraph isomorphism and homomorphism. In *LIPIcs-Leibniz International Proceedings in Informatics*, volume 47. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2016. (page 13, 51)

[KQS15] Raghav Kulkarni, Youming Qiao, and Xiaoming Sun. Any monotone property of 3-uniform hypergraphs is weakly evasive. *Theoretical Computer Science*, 588:16 – 23, 2015. (page 71)

[KS13] Raghav Kulkarni and Miklos Santha. Query complexity of matroids. In *Algorithms and Complexity*, pages 300–311. Springer, 2013. (page 7, 53, 68)

[KŠdW07] Hartmut Klauck, Robert Špalek, and Ronald de Wolf. Quantum and classical strong direct product theorems and optimal time-space tradeoffs. *SIAM Journal on Computing*, 36(5):1472–1493, 2007. (page 11, 104)

[KSS84] Jeff Kahn, Michael Saks, and Dean Sturtevant. A topological approach to evasiveness. *Combinatorica*, 4(4):297–306, 1984. (page 7, 68, 70)

[Kul13] Raghav Kulkarni. Evasiveness through a circuit lens. In *Proceedings of the 4th conference on Innovations in Theoretical Computer Science*, pages 139–144. ACM, 2013. (page 70, 71)

[LMR+11] Troy Lee, Rajat Mittal, Ben W Reichardt, Robert Spalek, and Mario Szegedy. Quantum query complexity of state conversion. In *Foundations of Computer Science (FOCS), 2011 IEEE 52nd Annual Symposium on*, pages 344–353. IEEE, 2011. (page 54)

[LMS11] Troy Lee, Frédéric Magniez, and Miklos Santha. A learning graph based quantum query algorithm for finding constant-size subgraphs. *arXiv preprint arXiv:1109.5135*, 2011. (page 53)

[LMT00] K.J. Lange, P. McKenzie, and A. Tapp. Reversible space equals deterministic space. *Journal of Computer and System Sciences*, 2(60):354–367, 2000. (page 114)

[LRR14] Yuan Li, Alexander Razborov, and Benjamin Rossman. On the ac0 complexity of subgraph isomorphism. In *Foundations of Computer Science (FOCS), 2014 IEEE 55th Annual Symposium on*, pages 344–353. IEEE, 2014. (page 52)

[Lut01] Frank H. Lutz. Some results related to the evasiveness conjecture. *Journal of Combinatorial Theory, Series B*, 81(1):110 – 124, 2001. (page 32, 70)

[Nay99] Ashwin Nayak. Optimal lower bounds for quantum automata and random access codes. In *Proceedings of 40th IEEE FOCS*, pages 369–376, 1999. quant-ph/9904093. (page 9, 88)

[NC10] Michael A Nielsen and Isaac L Chuang. *Quantum computation and quantum information*. Cambridge university press, 2010. (page 16, 26)

[Nec66] E. I. Neciporuk. A boolean function. In *Soviet Mathematics Doklady*, volume 7, 1966. (page 12, 111, 118, 119, 129)

[New91] Ilan Newman. Private vs. common random bits in communication complexity. *Information Processing Letters*, 39(2):67–71, 1991. (page 124)

[NS94] Noam Nisan and Mario Szegedy. On the degree of boolean functions as real polynomials. *Computational complexity*, 4(4):301–313, 1994. (page 36, 37, 38)

[NS96] Ilan Newman and Mario Szegedy. Public vs. private coin flips in one round communication games (extended abstract). In *Proceedings of the Twenty-eighth Annual ACM Symposium on Theory of Computing*, STOC '96, pages 561–570, 1996. (page 124)

[NW93] Noam Nisan and Avi Wigderson. Rounds in communication complexity revisited. *SIAM J. Comput.*, 22(1):211–219, February 1993. (page 122, 123)

[OSSS05] Ryan O'Donnell, Michael Saks, Oded Schramm, and Rocco A Servedio. Every decision tree has an influential variable. In *Foundations of Computer Science, 2005. FOCS 2005. 46th Annual IEEE Symposium on*, pages 31–39. IEEE, 2005. (page 3, 51)

[Pat92] Ramamohan Paturi. On the degree of polynomials that approximate symmetric boolean functions (preliminary version). In *Proceedings of the twenty-fourth annual ACM symposium on Theory of computing*, pages 468–474. ACM, 1992. (page 5, 41, 53, 55)

[PS84] Christos H. Papadimitriou and Michael Sipser. Communication complexity. *Journal of Computer and System Sciences*, 28(2):260–269, 1984. Earlier version in STOC'82. (page 11, 110)

[PSS14] Periklis A. Papakonstantinou, Dominik Scheder, and Hao Song. Overlays and limited memory communication mode(l)s. In *Proc. of the 29th Conference on Computational Complexity*, 2014. (page 11, 104)

[RS04] Ran Raz and Amir Shpilka. On the power of quantum proofs. In *Proceedings of Computational Complexity*, pages 260–274, 2004. (page 98, 99)

[RV76] Ronald L Rivest and Jean Vuillemin. On recognizing graph properties from adjacency matrices. *Theoretical Computer Science*, 3(3):371–384, 1976. (page 71)

[Sau72] Norbert Sauer. On the density of families of sets. *J. Combin. Theory Ser. A*, 13:145–147, 1972. (page 9, 88, 90)

[Ser14] I. S. Sergeev. Upper bounds for the formula size of symmetric boolean functions. *Russian Mathematics, Iz. VUZ*, 58(5):30–42, 2014. (page 119)

[Shi07] Yaoyun Shi. Approximate polynomial degree of boolean functions and its applications. In *Proc. of the 4th International Congress of Chinese Mathematicians*. Citeseer, 2007. (page 38)

[ST97] Rakesh Kumar Sinha and Jayram S Thathachar. Efficient oblivious branching programs for threshold and mod functions. *Journal of Computer and System Sciences*, 55(3):373–384, 1997. (page 115, 116)

[SY] Miklos Santha and Andrew Chi-Chih Yao. *Unpublished Manuscript*. (page 4, 51, 53, 57)

[Sym] *Wikipedia - Symmetric graph*. (page 6)

[SYZ04a] Xiaoming Sun, Andrew C Yao, and Shengyu Zhang. Graph properties and circular functions: How low can quantum query complexity go? In *Computational Complexity, 2004. Proceedings. 19th IEEE Annual Conference on*, pages 286–293. IEEE, 2004. (page 7, 68, 71)

[SYZ04b] Xiaoming Sun, Andrew C Yao, and Shengyu Zhang. Graph properties and circular functions: How low can quantum query complexity go? In *Computational Complexity, 2004. Proceedings. 19th IEEE Annual Conference on*, pages 286–293. IEEE, 2004. (page 59)

[Tur84] György Turán. The critical complexity of graph properties. *Information Processing Letters*, 18(3):151–153, 1984. (page 41)

[Wat00] John Watrous. Succinct quantum proofs for properties of finite groups. In *Proceedings of 41st IEEE FOCS*, pages 537–546, 2000. quant-ph/0011023. (page 10, 97)

[Wat06] John Watrous. *Lecture Notes On Introduction to Quantum Computing*, 2006. (page 16, 31)

[Wat11] John Watrous. *Lecture Notes On Theory of Quantum Information*, 2011. (page 16, 20)

[Weg87] Ingo Wegener. *The Complexity of Boolean Functions*. Wiley-Teubner Series in Computer Science, 1987. (page 12, 119, 120, 129)

[wik] *Wikipedia - Abel-Ruffini Theorem*. (page 52)

[Win04] Andreas Winter. Quantum and classical message identification via quantum channels. *arXiv preprint quant-ph/0401060*, 2004. (page 9, 88)

[Yao87] Andrew Chi-Chih Yao. Lower bounds to randomized algorithms for graph properties. In *Foundations of Computer Science, 1987., 28th Annual Symposium on*, pages 393–400. IEEE, 1987. (page 3, 51, 54)