A Generalized Adversary Method for Quantum Query Complexity

by

Rory Soiffer

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Mathematics
in
Computer Science

Waterloo, Ontario, Canada, 2022

Author's Declaration

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Abstract

Quantum query complexity measures the minimum number of queries a quantum algorithm needs to make to some input string to compute a function of that input. Query complexity models are widely used throughout quantum computing, from setting limits on quantum algorithms to analyzing post-quantum cryptography.

This thesis studies quantum adversary methods, a group of mathematical tools that prove lower bounds on quantum query complexity. I introduce a new general-purpose framework for adversary methods that generalizes over both the negative weight and multiplicative adversary methods. This framework unifies the lower bound proofs of both methods, even in the general case of quantum state conversion.

This generalized method also gives a new formula for the multiplicative adversary method based on max-relative entropy. This new definition is more concise and easier to reason about than existing definitions in the literature. I verify this by reproving several known results about the multiplicative adversary method. I also use this to reprove the strong direct product theorem for quantum query complexity.

Acknowledgements

I thank my advisor Shalev Ben-David, whose advice and support was critical to completing this thesis. His insight on the connection between max-relative entropy and the multiplicative adversary was the key to this entire research project.

I thank the University of Waterloo for its generous assistance, which made it possible for me to focus entirely on research during my degree.

I thank my parents, who inspired me to study computer science in the first place and who have always encouraged my passion for learning.

Dedication

I dedicate this thesis to my friends in Portland, Seattle, and elsewhere, who supported me through the trials of a remote degree program.

Table of Contents

Chapter 1

Introduction

When can a quantum computer outperform a classical computer? This question is the key to understanding the power and limitations of the entire field of quantum computing. The study of quantum complexity theory attempts to answer this question by comparing the optimal complexity of solving various problems on classical computers versus on quantum computers. By proving inequalities or bounds on the complexity of a problem, we prove whether it can be solved more efficiently on a quantum computer or not.

Query problems are one of the most-studied classes of problems within complexity theory. A query problem asks the following: given some function f and an input string x to that function, how many characters of x do I need to query to compute the value of $f(x)$? Many real-world algorithms can be modeled as query problems: database search is a famous example. The required number of queries depends on whether we have access to a classical computer or to a quantum computer. If we can only query one character at a time, we have classical query complexity. If we can query multiple characters in superposition, we have quantum query complexity. Proving separations between these two quantities is the main goal of the study of quantum query complexity.

Quantum query complexity is often analyzed through a group of tools called adversary methods. All adversary methods provide lower bounds on query complexity, and these lower bounds are even known to be tight in some cases. The study of adversary methods is a recent and evolving field. Early papers on adversary bounds presented several disconnected techniques [Zha05, BSS01, Amb06]. Then, [ŠS05] proved that all three methods were equivalent to a method now known as the positive weight adversary. Later, [HLS07] found an even stronger method (the negative weight adversary), and proved it was optimal for

constant-error quantum query complexity [LMRŠ10]. Most recently, [Špa08] found an even stronger method (the multiplicative adversary). While initially very complex, the multiplicative adversary has been simplified over time [AMRR11, MR15] and has been used to prove several important results in quantum query complexity, most notably the strong direct product theorem [LR12].

Each of three major adversary methods (positive weight, negative weight, and multiplicative) has strengths and weaknesses. The positive weight adversary is the easiest to use by far, and it has been used for most of the known practical bounds on actual query problems. The negative weight adversary is slightly more complex, but it has the major advantage of being a tight bound for constant-error query complexity. However, the negative weight adversary is not optimal for non-constant-error regimes. The multiplicative adversary is even more complex, but it produces a stronger bound than the negative weight adversary for these error regimes.

1.1 Goals

This thesis has three main goals:

1. To serve as an introduction to the field of quantum query complexity and adversary methods. This introduction is meant for readers who are familiar with quantum computing, but who are new to complexity theory. I focus on rigorously defining all the relevant terms from quantum information and query complexity, but I also try to motivate and give intuition behind those definitions.

2. To present a generalized adversary method that simplifies and unifies the correctness proofs of all existing adversary methods. This generalized method also gives us a framework for constructing new adversary methods. I show how we can use this framework to construct both the negative weight and multiplicative adversary methods.

3. To give an updated guide to the multiplicative adversary method. Using the generalized adversary method, I give a new formula for the multiplicative adversary based on max-relative entropy. I claim this new definition is simpler to work with than the standard definition. I verify this by collecting and reproving many existing results about the multiplicative adversary, including the strong direct product theorem for quantum query complexity.

1.2 Outline

The first half of this thesis covers background information necessary for the study of adversary methods. The material should be accessible to readers with a background in quantum computing. In Chapter 2, I present background information from quantum information that is necessary for the formal proofs in the rest of this thesis. In Chapter 3, I formally define all common forms of query complexity, both classical and quantum. In Chapter 4, I give background on the most important adversary methods used to prove lower bounds on query complexity.

In Chapter 5, I present a new generalized adversary method. This method generalizes over the concept of a progress function, such that each way of measuring the distance between two matrices yields a new lower bound for quantum query complexity. The negative weight adversary and the multiplicative adversary are just special cases of this generalized method. I also prove some lemmas that give further ways to construct special cases of this method.

In Chapter 6, I focus on the new form of the multiplicative adversary method given by this generalized method. This form is equal to the standard form used in the literature, but it is much simpler to express and to reason about. I reprove several simple results about the multiplicative adversary. In Chapter 7, I use this new form to reprove the strong direct product theorem for quantum query complexity.

Chapter 2

Overview of Quantum Information

Before formally defining query complexity and adversary methods, I give an overview of the relevant mathematical concepts from quantum information. The first half of this chapter reviews common terms in quantum information, such as positive matrices and Gram matrices. The second half of this chapter introduces less common definitions, many of which are specific to the study of particular adversary methods. Readers who are less familiar with quantum information are encouraged to read the whole chapter, while experienced readers may want to skip to Section 2.5.

I assume that readers have studied quantum computing before. This thesis makes heavy use of common terms such as mixed states or Hilbert spaces. I encourage readers who are unfamiliar with these terms to read an introduction to quantum computing, such as the excellent textbook by Nielsen and Chuang [NC10], before continuing.

2.1 Notation

I start by defining all the common notation I use in the rest of this thesis.

We work primarily on a register of n qubits. Let $\mathcal{H}$ be a complex Hilbert space of dimension $N = 2^n$, whose basis states are indexed by the bit strings of length n. For the rest of this thesis, I assume that all matrices and vectors are in $\mathcal{H}$ unless stated otherwise. Let $D(\mathcal{H})$ denote the set of mixed states on $\mathcal{H}$.

Since all of the proofs in this thesis make heavy use of matrices, I present my preferred notation for matrix norms and eigenvalues.

For any matrix X with real eigenvalues, let $\lambda(X)$ denote the vector of eigenvalues of X, sorted from largest to smallest. Let $\lambda_{max}(X)$ denote the largest eigenvalue of X, and let $\lambda_{min}(X)$ denote the smallest eigenvalue of X.

For any matrix X, let $\|X\|$ denote the spectral norm of X (the largest absolute value of any eigenvalue of X). Let $\|X\|_{tr}$ denote the trace norm of X (the sum of the absolute values of the eigenvalues of X).

For any matrices X, Y, I use the following notation:

- $X \circ Y$ denotes the Hadamard product of X and Y (also called the entrywise product or Schur product).
- $\langle X, Y \rangle$ denotes the Frobenius inner product (also called the entrywise inner product, equivalent to flattening X and Y to vectors and then taking the vector inner product).
- $X \geq Y$ means that $X - Y$ is a positive matrix (see below for a discussion of positive matrices).

2.2 Positive matrices

Positive matrices are fundamental to most of the proofs in this thesis. I first give the exact definitions I use, then give proofs of many of the basic properties satisfied by positive matrices.

Definition 2.1 (positive matrix)**.** A matrix is positive (also called positive semi-definite) if it is Hermitian and all of its eigenvalues are non-negative.

Definition 2.2 (positive cone)**.** $\text{Pos}(\mathcal{H})$ is the set of all positive matrices on the space $\mathcal{H}$.

Definition 2.3 (positive map)**.** A linear function Φ on the set of matrices is a positive map if it maps positive matrices to positive matrices.

In general, I allow positive matrices to have eigenvalues equal to 0. If having a zero eigenvalue would cause issues, such as when taking inverses, assume that the matrix has no zero eigenvalues. For the rest of this thesis, I ignore this special case for brevity.

There are several equivalent ways to define positive matrices.

Proposition 2.4. *The following statements are equivalent:*

1. *X is a positive matrix*

2. *$X = UDU^*$ for some unitary matrix U and some real diagonal matrix D with non-negative entries*
3. *$X = B^*B$ for some matrix B*
4. *X is Hermitian and $v^*Xv \geq 0$ for all vectors v*

Proof. (1) implies (2) by the spectral theorem. The diagonal elements of D are exactly the eigenvalues of X.

(2) implies (3), as we can let $B = D^{1/2}U^*$.

(3) implies (4), as B^*B is always Hermitian, and $v^*B^*Bv = (Bv)^*Bv = \|Bv\|^2$ is always non-negative.

(4) implies (1) by way of contradiction: if X had a negative eigenvalue λ, then its associated unit eigenvector would satisfy $v^*Xv = -\lambda$, contradicting (4). □

There are a myriad of ways to construct positive matrices:

- If X, Y are positive, then $X + Y$ is positive
- If X is positive and c is a non-negative real, then $c \cdot X$ is positive
- If X is positive and α is real, then X^α is positive
- If v is a vector, then vv^* is positive
- If X is Hermitian, then $\exp(X)$ is positive
- If X is positive and A is any matrix, then AXA^* is positive

Proposition 2.5 (Schur product theorem)**.** *If X, Y are positive matrices, then their Hadamard product $X \circ Y$ is also positive.*

Proof. By the spectral theorem, we can write $X = \sum_i v_i v_i^*$ and $Y = \sum_j u_j u_j^*$, where v_i and u_j are arbitrary vectors. Then we have

$$X \circ Y = (\sum_i v_i v_i^*) \circ (\sum_j u_j u_j^*) = \sum_{ij} v_i v_i^* \circ u_j u_j^* = \sum_{ij} (v_i \circ u_j)(v_i \circ u_j)^*$$

Since each $(v_i \circ u_j)(v_i \circ u_j)^*$ is positive, we know that $X \circ Y$ is a sum of positive matrices, so it is positive. □

Proposition 2.6. *If X, Y are positive matrices, then the product XY has all non-negative eigenvalues.*

Proof. We know that $Y^{1/2}XY^{1/2}$ is positive, so in particular, it has all non-negative eigenvalues. Since AB and BA share the same eigenvalues for any matrices A, B, we know that $XY^{1/2}Y^{1/2} = XY$ must have all non-negative eigenvalues as well. □

Note that XY is not necessarily positive, as it may not be Hermitian.

2.3 Gram matrices

One important special type of positive matrix is the set of Gram matrices. The definition of Gram matrices I use here differs slightly from the standard definition in the literature, I add the requirement that we have only unit vectors.

Definition 2.7 (Gram matrix)**.** Let $\{v_i\}$ be a list of N unit vectors, that are indexed by the bit strings of length n (just like the basis states of $\mathcal{H}$). Then the Gram matrix of $\{v_i\}$ is the matrix X defined by $\langle i|X|j\rangle = \langle v_i|v_j\rangle$, where $|i\rangle$ and $|j\rangle$ are basis states of $\mathcal{H}$.

In addition, I say that X is a Gram matrix if there exists any set of unit vectors $\{v_i\}$ whose Gram matrix is X.

The above definition motivates the use of Gram matrices for many cases. The definition below is equivalent and is often mathematically simpler to work with.

Proposition 2.8. *X is a Gram matrix if and only if X is a positive matrix with 1s along the diagonal.*

Proof. Let X be the Gram matrix of $\{v_i\}$. Let B be the (possibly not square) matrix whose i^{th} column is the vector v_i. Then we have $X = B^*B$, so X is positive. In addition, we have $\langle i|X|i\rangle = \langle v_i|v_i\rangle = \|v_i\|^2 = 1$, so X has 1s along the diagonal.

For the converse, let X be a positive matrix with 1s along the diagonal. Let B be a square matrix such that $X = B^*B$. Then let v_i be the i^{th} column of B. This implies that $\langle i|X|j\rangle = \langle Bi|Bj\rangle = \langle v_i|v_j\rangle$. In addition, we have $1 = \langle i|X|i\rangle = \langle v_i|v_i\rangle = \|v_i\|^2$, so each v_i is a unit vector. □

Many common matrices are examples of Gram matrices. For instance:

- $\mathbb{I}$, the identity matrix, is a Gram matrix

- J, the all 1s matrix, is a Gram matrix
- If v is a vector whose entries all have magnitude 1, then vv^* is a Gram matrix

The set of Gram matrices is also closed under convex combinations and under Hadamard products:

- If X, Y are Gram matrices and $c \in [0, 1]$, then $cX + (1 - c)Y$ is a Gram matrix
- If X, Y are Gram matrices, then $X \circ Y$ is a Gram matrix

Also note that if ρ is a mixed state and X is a Gram matrix, then $\rho \circ X$ is a mixed state.

2.4 Fidelity

Fidelity is a common tool for measuring the similarity between a pair of mixed states. Following [MR15], I also define the Hadamard product fidelity, which measures the similarity between a pair of Gram matrices. Hadamard product fidelity is used as part of the output condition for ϵ-error query complexity.

I use the standard definition of quantum fidelity. This is the square of the fidelity used in [NC10], [LR12], [MR15], and others, which I refer to as square-root fidelity.

Definition 2.9 (fidelity)**.** For any mixed states $\rho, \sigma \in D(\mathcal{H})$, define

$$F(\rho, \sigma) = \left\| \rho^{1/2} \sigma^{1/2} \right\|_{tr}^2$$

Definition 2.10 (Hadamard product fidelity)**.** For any Gram matrices ρ, σ, define

$$F_H(\rho, \sigma) = \min_{A \in D(\mathcal{H})} F(\rho \circ A, \sigma \circ A)$$

Note that both fidelity and Hadamard product fidelity always return values in the range $[0, 1]$. Both functions return 1 only when the two inputs are equal.

2.5 Max-relative entropy

Max-relative entropy is another way to measure the distance between a pair of matrices. It is not a metric, and it is not even symmetric. However, it satisfies a number of useful properties when applied to positive matrices.

Max-relative entropy is not the same thing as relative entropy. As a side note, the two quantities are connected: if you generalize relative entropy to the order-α Renyi divergence, then you get max-relative entropy as you take the limit $\alpha \to \infty$ [MLDS+13]. This thesis doesn't use normal relative entropy or Renyi entropy.

Definition 2.11 (max-relative entropy)**.** Let X, Y be positive matrices. Then the max-relative entropy of X relative to Y is

$$\mathrm{D}_{\max}(X\|Y) = \inf\{\lambda \in \mathbb{R} : X \leq \exp(\lambda)Y\}$$

Since we're taking an infimum, if there is no λ for which $X \leq \exp(\lambda)Y$, then we define $\mathrm{D}_{\max}(X\|Y) = \infty$. If $X \leq \exp(\lambda)Y$ for all $\lambda \in \mathbb{R}$, then we define $\mathrm{D}_{\max}(X\|Y) = -\infty$.

Furthermore, in some cases, it is useful to extend the domain of $\mathrm{D}_{\max}$ to the case where X is not a positive matrix. All the properties below will still hold so long as X has at least one non-negative eigenvalue.

There are a number of equivalent definitions of the max-relative entropy:

1. $\mathrm{D}_{\max}(X\|Y) = \log \inf\{\lambda \in \mathbb{R}^+ : X \leq \lambda Y\}$
2. $\mathrm{D}_{\max}(X\|Y) = \log \left\|Y^{-1/2} X Y^{-1/2}\right\|$
3. $\mathrm{D}_{\max}(X\|Y) = \log \lambda_{max}(XY^{-1})$
4. $\mathrm{D}_{\max}(X\|Y) = \log \sup_{Z \geq 0} \frac{\langle X, Z\rangle}{\langle Y, Z\rangle}$

Most of these definitions are rearrangements of the terms. Item (4) in the list is slightly more complex to prove: it can be derived by noting that $X \leq \lambda Y$ if and only if $\langle X, Z\rangle \leq \lambda\langle Y, Z\rangle$ for all $Z \in \mathrm{Pos}(X)$.

I define two more quantities that are closely related to $\mathrm{D}_{\max}$ and that are used by several proofs later in this thesis.

Definition 2.12. Let X, Y be positive matrices. Then I define

$$R(X\|Y) = \exp(\mathrm{D}_{\max}(X\|Y)) = \inf\{\lambda \in \mathbb{R}^+ : X \leq \lambda Y\}$$

$$\delta_\infty(X, Y) = \max\{\mathrm{D}_{\max}(X\|Y), \mathrm{D}_{\max}(Y\|X)\}$$

The quantity R gets rid of an extra log, which simplifies some of the proofs. The quantity δ_∞ represents a symmetrized $\mathrm{D}_{\max}$, and it turns out to be a valid metric on $\mathrm{Pos}(\mathcal{H})$.

Max-relative entropy satisfies the following properties for all positive X, Y:

- $\mathrm{D}_{\max}(X\|Y) \leq 0$ if and only if $X \leq Y$ (follows from definition 1)
- If $\operatorname{Tr} X = \operatorname{Tr} Y$ (such as when X, Y are both mixed states or both Gram matrices), then $\mathrm{D}_{\max}(X\|Y) \geq 0$ (follows from the above property)
- If vv^*, uu^* are distinct pure states, then $\mathrm{D}_{\max}(vv^*\|uu^*) = \infty$ (follows from definition 4, letting Z be orthogonal to uu^*)
- $\mathrm{D}_{\max}(X\|Y) = \mathrm{D}_{\max}(Y^{-1}\|X^{-1})$ (follows from definition 3)
- $\mathrm{D}_{\max}(X\|\mathbb{I}) = \log\|X\|$ (follows from definition 2)

I give proofs of several interesting and useful properties of $\mathrm{D}_{\max}$.

Proposition 2.13 (quasiconvexity)**.** *If X_i, Y_i are all positive, then*

$$\mathrm{D}_{\max}(\sum_i X_i\|\sum_i Y_i) \leq \max_i \mathrm{D}_{\max}(X_i\|Y_i)$$

Proof. I show an equivalent result, that $R(\sum_i X_i\|\sum_i Y_i) \leq \max_i R(X_i\|Y_i)$. This implies the desired result if you take the log of both sides.

Let $M = \max_i R(X_i\|Y_i)$. Then we know that $X_i \leq MY_i$ for all i. This implies that $\sum_i X_i \leq \sum_i MY_i = M\sum_i Y_i$. Therefore $R(\sum_i X_i\|\sum_i Y_i) \leq M$. □

Proposition 2.14. *If X, Y are positive and Φ is a positive map, then* $\mathrm{D}_{\max}(\Phi(X)\|\Phi(Y)) \leq \mathrm{D}_{\max}(X\|Y)$

Proof. Let λ be the smallest real such that $X \leq \exp(\lambda)Y$. Then we know that $\exp(\lambda)Y - X \geq 0$, and since Φ maps positive matrices to positive matrices, we have $\Phi(\exp(\lambda)Y - X) \geq 0$. Since Φ is linear, we have $\exp(\lambda)\Phi(Y) - \Phi(X) \geq 0$, so $\Phi(X) \leq \exp(\lambda)\Phi(Y)$. Thus λ is also a valid solution to $\mathrm{D}_{\max}(\Phi(X)\|\Phi(Y))$, so we conclude that $\mathrm{D}_{\max}(\Phi(X)\|\Phi(Y)) \leq \mathrm{D}_{\max}(X\|Y)$. □

Corollary 2.15. *If X, Y, Z are positive, then* $\mathrm{D}_{\max}(X \circ Z\|Y \circ Z) \leq \mathrm{D}_{\max}(X\|Y)$.

Proof. We know by Proposition 2.5 that $\Phi(\rho) = \rho \circ Z$ is a positive map. Therefore Proposition 2.14 applies and gives us the desired result. □

Proposition 2.16. *If X, Y are positive and A is invertible, then* $\mathrm{D}_{\max}(AXA^*\|AYA^*) = \mathrm{D}_{\max}(X\|Y)$.

Proof. We know that $\Phi(\rho) = A\rho A^*$ is a positive map. By Proposition 2.14, we have

$$\mathrm{D}_{\max}(AXA^* \| AYA^*) \leq \mathrm{D}_{\max}(X\|Y)$$

The same argument works for A^{-1}, giving us

$$\mathrm{D}_{\max}(A^{-1}X'A^{-1^*} \| A^{-1}Y'A^{-1^*}) \leq \mathrm{D}_{\max}(X'\|Y')$$

Substituting $X' = AXA^*$ and $Y' = AYA^*$ gives $\mathrm{D}_{\max}(X\|Y) \leq \mathrm{D}_{\max}(AXA^*\|AYA^*)$. Combining this with the above completes the theorem. □

Proposition 2.17. *If X, Y, Z are positive, then* $\mathrm{D}_{\max}(X\|Z) \leq \mathrm{D}_{\max}(X\|Y) + \mathrm{D}_{\max}(Y\|Z)$.

Proof. I show an equivalent result, that $R(X\|Z) \leq R(X\|Y) \cdot R(Y\|Z)$. This implies the desired result if you take the log of both sides.

Let $\lambda_1 = R(X\|Y)$ and $\lambda_2 = R(Y\|Z)$. Then we have $X \leq \lambda_1 Y$ and $Y \leq \lambda_2 Z$, and combining these gives $X \leq \lambda_1\lambda_2 Z$. Therefore $R(X\|Z) \leq \lambda_1\lambda_2$, completing the proof. □

Proposition 2.18. *If X, Y, Z, W are positive, then* $\mathrm{D}_{\max}(X \otimes Y\|Z \otimes W) = \mathrm{D}_{\max}(X\|Z) + \mathrm{D}_{\max}(Y\|W)$.

Proof. By definition 3, we have

$$\mathrm{D}_{\max}(X \otimes Y\|Z \otimes W) = \log \lambda_{max}((X \otimes Y)(Z \otimes W)^{-1}) = \log \lambda_{max}(XZ^{-1} \otimes YW^{-1})$$

Since the eigenvalues of a tensor product are the products of the eigenvalues, this equals

$$\log\left(\lambda_{max}(XZ^{-1}) \cdot \lambda_{max}(YW^{-1})\right) = \log \lambda_{max}(XZ^{-1}) + \log \lambda_{max}(YW^{-1})$$

which is exactly equal to $\mathrm{D}_{\max}(X\|Z) + \mathrm{D}_{\max}(Y\|W)$. □

2.6 Factorization norm

For one of the proofs in Chapter 7, I use a somewhat obscure quantity called the factorization norm or the γ_2 norm. This norm was originally introduced to quantum complexity theory by [LMSS07], which related it to several existing measures in complexity theory. The γ_2 norm can be defined as follows.

Definition 2.19 (factorization norm)**.** Let A be a matrix. Let $c(Y)$ denote the largest L_2 norm of a column of Y. Then define

$$\gamma_2(A) = \min_{X,Y:X^*Y=A} c(X)c(Y)$$

[LSŠ08] is a good reference for properties of the γ_2 norm. This thesis only uses a couple of those properties. First, that the term factorization norm is justified, as γ_2 is a norm on the set of matrices. Second, that for any matrix A, we have

$$\gamma_2(A) = \max_{Q\neq 0} \frac{\|A \circ Q\|}{\|Q\|} = \max_{Q\neq 0} \frac{\|A \circ Q\|_{tr}}{\|Q\|_{tr}}$$

I prove one additional property of the γ_2 norm, that it returns 1 for all Gram matrices.

Proposition 2.20. *If σ is a Gram matrix, then $\gamma_2(\sigma) = 1$.*

Proof. Since σ is a Gram matrix, we can write $\sigma = B^*B$, where each of the columns of B has norm 1. Plugging this into Definition 2.19, we get that $\gamma_2(\sigma) \leq 1$.

Let Q be an arbitary mixed state. Note that $\|Q\|_{tr} = 1$, and $\sigma \circ Q$ is also a mixed state, so $\|\sigma \circ Q\|_{tr} = 1$ as well. Then by the property above, we have $\gamma_2(\sigma) \geq 1$. □

Chapter 3

Overview of Query Complexity

This chapter introduces classical and quantum query complexity. While parts of this chapter are intended for readers who are new to quantum complexity theory, even readers familiar with the field may want to read through the notation and precise definitions, as I make heavy use of them in my later proofs.

3.1 Motivation

Query problems are an important type of problem in complexity theory. In a query problem, the algorithm is given a hidden input string, and it can only read the values of this string by querying one position at a time. The goal of the algorithm is to compute some function of the input string while making as few queries as possible. The query complexity of the problem measures the minimum number of queries required to exactly compute the desired function for any input string.

Note that query complexity is not the same as time complexity, the number of elementary operations it takes to run an algorithm. While time complexity is arguably more useful in practice, it depends significantly on the set of elementary operations available to the system, and it can be almost impossible to compute theoretically. Query complexity can act as a good proxy for time complexity: it often relates closely to time complexity, while being much simpler to compute.

Query problems occur naturally in a variety of settings. For example, searching an unsorted array is a query problem. The input string is the array to search, and each query checks whether one element in the array matches the desired element. It is well-known that the

classical query complexity of searching an unsorted array of length n is $O(n)$, as in the worst case, you need to check every element of the array in order to find the desired element. A quantum computer can solve this problem faster with Grover's algorithm, which only requires $O(\sqrt{n})$ queries to the input [NC10].

Query complexity has many applications throughout quantum computing, to list a few:

1. Shor's algorithm depends on the order finding subproblem, which can be naturally expressed as a query problem [Amb18].
2. Quantum walk algorithms are often used to solve query problems, so bounding query complexity also bounds quantum walks [San08].
3. Communication complexity can be reduced to query complexity, giving a simple way to prove bounds in that field [BCW98].
4. Oracles in complexity theory are just a type of query problem, so query complexity can be used to prove oracle separations, such as the one between BQP and PH [RT19].
5. Post-quantum cryptography uses quantum oracles as a hardness model, so it uses results from query complexity [BDF+11].

To find the query complexity of a problem, we need to prove matching upper and lower bounds for query complexity. It is generally easy to find upper bounds for query complexity: any algorithm that solves the problem is an upper bound on the query complexity. Thus, the study of query complexity usually focuses on techniques that prove lower bounds for query complexity, such as the adversary methods discussed in the next chapter.

3.2 Classical query complexity

I formalize the notion of query complexity motivated above. Recall that the goal of a query problem is to compute the output of some function f while making as few queries to the input of f as possible. Let f be a function $X^n \to Y$, where X and Y are arbitrary finite sets. We often use $X = \{0, 1\}$, so that the input is a bit string of length n. In this thesis, I assume f is a total function (that is, f is defined on its entire domain). Other papers on query complexity often allow f to be a partial function, where the query problem assumes the input is in a given subset of the domain.

Let $x \in X^n$ be the input to f. The queries to x take the form of an oracle $O_x : [n] \to X$ that returns the i^{th} element of x. In other words, O_x is defined by $O_x(i) = x_i$. We want to write an algorithm A that computes $f(x)$ while making as few calls to O_x as possible.

Our algorithm is not allowed to access the input x directly, but it may perform any other operations it wants.

Definition 3.1 (deterministic query complexity)**.** The deterministic query complexity $D(f)$ is the minimum number of calls to O_x that the best deterministic algorithm A must make to be guaranteed to solve the problem on any input x.

Definition 3.2 (randomized query complexity)**.** The randomized query complexity $R(f)$ is the minimum expected number of calls to O_x that the best randomized algorithm A must make to solve the problem on any input x.

Since randomized algorithms are a superset of deterministic algorithms, we know that $R(f) \leq D(f)$ for all f. For total functions, it is known that $D(f) \leq O(R(f)^2)$ [Amb18].

3.3 Quantum query complexity

Quantum query complexity is defined similarly to classical query complexity. The key difference is that we can now query the input in superposition: O_x changes from a classical query function to a quantum oracle unitary.

For simplicity, I assume that $X = \{0, 1\}$, so our input is a bit string. Then O_x is an operator on a Hilbert space $\mathcal{H}_{\text{query}}$ with basis states indexed by $i \in [n]$. The action of O_x is defined by $O_x |i\rangle = (-1)^{x_i} |i\rangle$. Since $x_i \in \{0, 1\}$, O_x either flips the phase or has no effect on the state. Assuming we are able to perform controlled O_x gates (such as by adding an extra index i' such that $x_{i'}$ is always 0), we can transform this phase flip into a bit flip and determine the value of x_i. O_x could equivalently be defined through a bit flip instead of a phase flip; I choose this definition because it doesn't require us to add an extra register. The generalization to $X \neq \{0, 1\}$ would require we add this extra register, as a phase flip can't represent more than a bit of information.

As before, we want to write an algorithm A that computes $f(x)$ while making as few calls to O_x as possible. Unlike before, A is now a quantum circuit. Our circuit is not allowed to access the input x directly, but it may perform any other unitary operations it wants. We can assume without loss of generality that A has no measurements (as we can always delay measurements until the end). All the registers of A are assumed to start in the $|0\rangle$ state.

The output of A is stored in a register $\mathcal{H}_{\text{ans}}$ with basis states indexed by Y. There are two ways to define whether A yields the correct output, which give two different definitions for quantum query complexity:

Definition 3.3 (quantum query complexity)**.** The (exact) quantum query complexity $Q(f)$ is the minimum number of calls to O_x that the best quantum algorithm A must make to be guaranteed to solve the problem on any input x, with no chance of error. We say that A is correct if the register $\mathcal{H}_{\text{ans}}$ ends in the basis state $|f(x)\rangle$ (after we trace out all the other registers).

Definition 3.4 (coherent quantum query complexity)**.** The coherent quantum query complexity $Q^c(f)$ is the same as $Q(f)$, except that we say that A is correct if the register $\mathcal{H}_{\text{ans}}$ ends in the basis state $|f(x)\rangle$ **and** all the other registers end in the $|0\rangle$ state.

Coherent quantum query complexity is stricter than non-coherent quantum query complexity, so we know that $Q(f) \leq Q^c(f)$. It is known that $Q^c(f) \leq 2Q(f)$ [LR12], as we can run the algorithm twice to uncompute any non-zero values in the ancillary registers. Coherent quantum query complexity is useful when we need to guarantee that our algorithm won't collapse the state, such as if we are using the algorithm as a subroutine for a larger quantum algorithm. The stricter output condition can also make it easier to reason about.

Since quantum algorithms are a superset of randomized algorithms, we know that $Q(f) \leq R(f)$ for all f. Surprisingly, for total functions, it is known that $R(f) \leq O(Q(f)^3)$ [Amb18]. Thus, for total functions, quantum computers can never achieve more than a polynomial speedup over classical computers.

For the proofs in Chapter 5, it is helpful to have a standard mathematical representation of quantum query algorithms. A query algorithm A is allowed to perform two types of operations: it can call unitary gates and it can call the query oracle. Since any composition of unitaries yields another unitary, we can assume without loss of generality that A calls exactly one unitary in between every pair of calls to the query oracle. Thus, if A calls O_x a total of T times, A takes the form

$$A = U_T O_x U_{T-1} \dots U_1 O_x U_0$$

where each U_t is an arbitrary unitary operator on some Hilbert space $\mathcal{H}$. We know that $\mathcal{H}$ must contain $\mathcal{H}_{\text{query}}$ and $\mathcal{H}_{\text{ans}}$, but it may also contain any number of ancillary registers.

3.4 Quantum state generation and conversion

The previous section generalized query complexity to allow querying bits in superposition. The natural next step is to generalize query complexity to allow quantum outputs. This

means that Y no longer represents a set of classical outputs, but Y represents a set of quantum states.

Let Y be a set of pure states on a Hilbert space $\mathcal{H}_{\text{ans}}$. Let $f(x) = |\psi_x\rangle \in \mathcal{H}_{\text{ans}}$ be the desired output state on an input x. The definitions from the section above still apply to this case, except that instead of trying to output a basis state of $\mathcal{H}_{\text{ans}}$, we want to output $|\psi_x\rangle$.

Definition 3.5 (quantum query complexity of state generation)**.** The quantum query complexity of state generation $Q(f)$ is identical to the quantum query complexity, except that the output condition checks for the pure state $f(x)$ instead of the basis state $|f(x)\rangle$.

It can be difficult to directly reason about a function that outputs pure states. We can simplify the problem by observing that the query complexity of f is entirely determined by the inner products $\langle\psi_x|\psi_y\rangle$ for each $x, y \in X^n$. This means that we only need to know the Gram matrix of $\{|\psi_x\rangle\}_x$, we don't need to know the exact values of each $|\psi_x\rangle$.

Proposition 3.6. *Let f_1, f_2 be functions $X^n \to \mathcal{H}_{ans}$, and let $f_1(x) = |\psi_x\rangle$ and $f_2(x) = |\phi_x\rangle$ be pure states. Assume that the Gram matrix of $\{|\psi_x\rangle\}_x$ is σ and the Gram matrix of $\{|\phi_x\rangle\}_x$ is also σ. Then we have $Q(f_1) = Q(f_2)$.*

Proof. I claim there is a unitary U such that $U|\psi_x\rangle = |\phi_x\rangle$ for all x. This follows from the fact that if $B_1 B_1^* = B_2 B_2^*$, then there exists a unitary U such that $UB_1 = B_2$.

Let A_1 be the best algorithm that solves the query problem for f_1, so that A_1 makes $Q(f_1)$ calls to O_x. Then, let $A_2 = UA_1$. A_2 is a valid quantum circuit, and it solves the query problem for f_2 in $Q(f_1)$ queries, so we conclude that $Q(f_2) \leq Q(f_1)$. By a symmetric argument, we have $Q(f_1) \leq Q(f_2)$, so we must have $Q(f_1) = Q(f_2)$. □

If $f(x) = |\psi_x\rangle$ and the Gram matrix of $\{|\psi_x\rangle\}_x$ is σ, then I define $Q(\sigma) = Q(f)$ and $Q^c(\sigma) = Q^c(f)$. We know that $Q(\sigma)$ and $Q^c(\sigma)$ are well-defined by the proposition above. Since Gram matrices are often very easy to work with, I primarily use this Gram matrix notation for the rest of this thesis.

Note that this Gram matrix notation applies whether f outputs a quantum state or a classical value (as in the section above). If f outputs a classical value, then we take the Gram matrix of the associated basis states. Since the inner product of two distinct basis states is always 0, we know that $\langle x|\sigma|y\rangle = 0$ whenever $f(x) \neq f(y)$, and $\langle x|\sigma|y\rangle = 1$ whenever $f(x) = f(y)$.

As another generalization to quantum query complexity, I consider the case where our register doesn't necessarily start in the $|0\rangle$ state for every input x. Instead, we have a function $f_1 : X^n \to \mathcal{H}_{\text{ans}}$ that specifies the initial state of our register given an input x, alongside the function f_2 that specifies the desired final state of the register.

Definition 3.7 (quantum query complexity of state conversion)**.** The quantum query complexity of state conversion $Q(f_1, f_2)$ is identical to the quantum query complexity of state generation, except that the register $\mathcal{H}_{\text{ans}}$ starts in the state $f_1(x)$ on an input x.

As above, the query complexity only depends on Gram matrices associated with f_1 and f_2, and it's easier to work with Gram matrices than with functions. If the Gram matrix of $\{f_1(x)\}_x$ is σ_1 and the Gram matrix of $\{f_2(x)\}_x$ is σ_2, then I define $Q(\sigma_1, \sigma_2) = Q(f_1, f_2)$.

Note that state conversion is a generalization of state generation. In state generation, the register $\mathcal{H}_{\text{ans}}$ starts in the $|0\rangle$ state regardless of the input x, which means that $f_1(x) = |0\rangle$, so we have $\sigma_1 = J$. Thus we have $Q(\sigma) = Q(J, \sigma)$.

3.5 ϵ-error query complexity

The above sections all assumed that we wanted to exactly compute the output of a function, with no chance of outputting an incorrect result. In practice, a small amount of error is often acceptable, and can make the problem much easier to solve. All of the above notions of query complexity (except deterministic query complexity) can be generalized to accept algorithms that return incorrect results with probability ϵ.

Definition 3.8 (ϵ-error randomized query complexity)**.** The ϵ-error randomized query complexity $R_\epsilon(f)$ is the minimum expected number of calls to O_x that the best randomized algorithm A must make to solve the problem on any input x with probability at least $1 - \epsilon$.

For quantum query complexity, we output a mixed state, so we can't check if it matches a given pure state with some probability. Instead, we check that the fidelity of our output state with our goal state is at least some threshold.

Definition 3.9 (ϵ-error quantum query complexity)**.** The ϵ-error quantum query complexity $Q_\epsilon(f)$ is the minimum number of calls to O_x that the best quantum algorithm A must make to be guaranteed to solve the problem on any input x. We say that A is correct if the register $\mathcal{H}_{\text{ans}}$ ends in a state ρ (after we trace out all the other registers) such that $F(\rho, |f(x)\rangle \langle f(x)|) \geq 1 - \epsilon$.

ϵ-error versions of the other forms of quantum query complexity (coherent, state generation,

and state conversion) are defined analogously.

We can simplify the definition of ϵ-error quantum query complexity with the following trick. First, note that any quantum algorithm A will always produce the same output mixed state ρ_x for the same input x. Thus, A solves the exact state generation problem given by $f'(x) = \rho_x$ (where we generalize state generation to allow mixed states, redefining the resulting Gram matrix as $\sigma[x, y] = \sqrt{F(\rho_x, \rho_y)}$). This means that we can express ϵ-error quantum query complexity in terms of exact quantum query complexity just by expressing the output condition as a relationship between f' and f.

Proposition 3.10. *For any Gram matrix σ, we have*

$$Q_\epsilon(\sigma) = \min_\rho Q(\rho)$$

where ρ ranges over all Gram matrices with $F_H(\rho, \sigma) \leq 1 - \epsilon$.

Proof. This result is shown in [MR15, LR12]. Note that those papers use square-root fidelity, so their statement different slightly from the one above. □

The other forms of quantum query complexity (coherent, state generation, and state conversion) satisfy analogous properties.

Finally, note that if ϵ is a positive constant, then the exact value of ϵ is often not very important. In many cases, we can run the algorithm multiple times and collect the outputs together to reduce error. For example, for randomized query complexity, we can simply run the algorithm k times and take the majority output to reduce the error to roughly ϵ^k. Similar procedures exist for the quantum query complexity of a classical function (though not for arbitrary state generation problems). These procedures let us decrease ϵ to any desired constant value with only a constant factor of overhead. Therefore, ignoring constant factors, the query complexity is the same for any constant value of ϵ. This quantity is called constant-error or bounded-error query complexity.

Chapter 4

Overview of Adversary Methods

As discussed above, the main goal of the study of query complexity is to prove lower bounds for query complexity. This chapter introduces adversary methods, which are a class of techniques that prove lower bounds for quantum query complexity.

All adversary methods share a similar form: they let Γ be a special type of matrix, called an adversary matrix, and they maximize over some function of Γ. The set of valid adversary matrices changes for each method, as does the function being maximized.

4.1 Positive weight adversary method

Quantum adversary methods were first introduced in [Amb02]. This paper proved lower bounds on quantum query complexity by imagining an adversary that tries to trick the algorithm into producing the wrong solution. Roughly, this works by first picking some input to the problem, then changing the input slightly in a way that changes the desired output. Since the algorithm is limited in the number of bits it can query, it is limited in its ability to detect the small change in the input, so it must produce similar outputs on both inputs. Since the two inputs have different desired outputs, the algorithm must be wrong on one of them, completing the lower bound. I encourage readers to refer to the original paper for details.

This method was generalized in [Amb06] to allow for weighted connections between pairs of inputs. Very roughly, a single weight corresponds to the probability that the adversary changes the first input to the second input. Since there are N possible inputs, the weights form a matrix of dimension $N \times N$, called the adversary matrix.

As a followup, [ŠS05] proved this method was equivalent to a number of other lower bound techniques known in the literature. The two most important equivalent forms in that paper are the spectral adversary and weighted adversary, which are defined below.

The following matrices are used in the definition of the adversaries below.

Definition 4.1 (special matrices)**.** For $i \in [n]$, define matrices D_i and S_i by

$$D_i[x,y] = \begin{cases} 0 & x_i = y_i \\ 1 & x_i \neq y_i \end{cases} \qquad S_i[x,y] = \begin{cases} 1 & x_i = y_i \\ -1 & x_i \neq y_i \end{cases}$$

Note that $D_i = (J - S_i)/2$ and that S_i is a rank 1 Gram matrix, as $S_i = v_i v_i^*$ where $v_i[x] = (-1)^{x_i}$. While many of the existing definitions in the literature use D_i, I find that S_i is more elegant to use in my generalized theorems.

Definition 4.2 (spectral adversary)**.** Let f be a classical function with Gram matrix F. Let Γ be a real symmetric matrix with non-negative entries such that $\Gamma \circ F = 0$. Then define

$$\mathrm{SA}(f) = \sup_{\Gamma} \frac{\|\Gamma\|}{\max_i \|\Gamma \circ D_i\|}$$

Definition 4.3 (weighted adversary)**.** Let f be a classical function. Let $w : X^n \times X^n \to \mathbb{R}$ and $w' : X^n \times X^n \times [n] \to \mathbb{R}^+$ be functions that satisfy the following properties:

- $w(x,y) = w(y,x)$
- If $f(x) = f(y)$, then $w(x,y) = 0$
- If $x_i = y_i$ or $f(x) = f(y)$, then $w'(x,y,i) = 0$
- If $x_i \neq y_i$ and $f(x) \neq f(y)$, then $w'(x,y,i)w'(y,x,i) \geq w^2(x,y)$

Then define $wt(x) = \sum_y w(x,y)$ and $v(x,i) = \sum_y w'(x,y,i)$, and finally define

$$\mathrm{WA}(f) = \max_{w,w'} \min_{x,y,i,j} \sqrt{\frac{wt(x)wt(y)}{v(x,i)v(y,j)}}$$
$$\text{subject to } f(x) \neq f(y)$$
$$v(x,i)v(y,j) > 0$$

Theorem 4.4. *Let f be a classical function. Then* $\mathrm{SA}(f) = \mathrm{WA}(f)$.

Proof. See Theorem 3.1 of [ŠS05]. □

In recent literature, this quantity is usually referred to as the positive weight adversary, to distinguish it from the negative weight adversary below. For the rest of this thesis, I mostly use the spectral adversary definition, as it's shorter and it uses adversary matrices in the same way as the other adversary methods. In practice many people prefer the weighted adversary definition, as it's usually easier to find weight schemes w, w' than it is to compute the spectral norm of an adversary matrix Γ.

4.2 Negative weight adversary method

While the positive weight adversary is relatively intuitive and easy to use, it is not a tight lower bound on quantum query complexity. The first step in strengthening that bound was in [HLS07], which generalized the spectral adversary above to allow the adversary matrix to have positive or negative entries.

Definition 4.5 (negative weight adversary)**.** Let f be a classical function with Gram matrix F. Let Γ be a real symmetric matrix such that $\Gamma \circ F = 0$. Then define

$$\mathrm{Adv}^{\pm}(f) = \sup_{\Gamma} \frac{\|\Gamma\|}{\max_i \|\Gamma \circ D_i\|}$$

The only difference between $\mathrm{Adv}^{\pm}$ and SA is that we remove the restriction that all the entries of Γ are non-negative. Since we are taking a supremum over Γ, we immediately get $\mathrm{Adv}^{\pm}(f) \geq \mathrm{SA}(f)$.

The same paper proved $Q(f) \geq \frac{1}{2}\mathrm{Adv}^{\pm}(f)$, so the negative weight adversary is a lower bound for zero-error quantum query complexity. Similar results are known for ϵ-error quantum query complexity, though the constant factor depends on the value of ϵ [HLS07]. By the inequality above, these lower bounds also hold for the positive weight adversary.

Surprisingly, this stronger adversary method turns out to be a tight bound for constant-error quantum query complexity.

Theorem 4.6. *Let f be a classical function. Then*

$$Q_{1/3}(f) = \Theta(\mathrm{Adv}^{\pm}(f))$$

The choice of the constant $\frac{1}{3}$ is arbitrary, the theorem holds for any other constant in $(0, \frac{1}{2})$.

Proof. See Theorem 1.1 of [LMRŠ10]. □

The negative weight adversary thus gives us the exact value of the bounded-error quantum query complexity of any function (up to a constant factor). Since bounded-error quantum query complexity is the most practically useful form of query complexity, the negative weight adversary is ideal for most cases. However, the negative weight adversary is not tight for all forms of quantum query complexity. For zero-error quantum query complexity, or if the error ϵ approaches 0 or $\frac{1}{2}$ in the limit, the negative weight adversary is only a lower bound. It has been shown that the negative weight adversary is not always tight (up to a constant factor) in those error regimes. Thus, we may be able to find even stronger adversary methods that outperform the negative weight adversary method in those cases.

4.3 Multiplicative adversary method

One adversary method that can outperform the negative weight adversary method in these non-constant-error regimes is the multiplicative adversary method. First introduced in [Špa08], this method works by bounding a multiplicative error between subsequent calls to the oracle as opposed to an additive error between subsequent calls to the oracle, as in the negative weight adversary. For this reason, the negative weight adversary is sometimes called the additive adversary, to distinguish it from the multiplicative adversary.

The multiplicative adversary method is usually defined as an optimization problem that almost takes the form of a semi-definite program. Different papers give slightly different definitions, I present the definition given in [MR15].

Definition 4.7 (multiplicative adversary)**.** Let σ be a Gram matrix. Then define

$$\begin{aligned} \mathrm{Madv}(\sigma) = \sup_{c>1} \frac{1}{\log c} \quad & \max_{\Gamma \in \mathrm{Pos}(\mathcal{H})} && \log \mathrm{Tr}(\Gamma\sigma) \\ & \text{subject to} && \mathrm{Tr}(\Gamma J) = 1 \\ & && \Gamma \circ S_i \leq c\Gamma \text{ for all } i \in [n] \end{aligned}$$

The multiplicative adversary is a lower bound on quantum query complexity of state generation, we know that $Q^c(\sigma) \geq \mathrm{Madv}(\sigma)$ [LR12]. We also know that for a classical function f with Gram matrix F, $\mathrm{Madv}(F) = \Omega(\mathrm{Adv}^{\pm}(f))$ (see [AMRR11] or Theorem 6.4), so the multiplicative adversary is strictly stronger than the negative weight adversary. Unlike the negative weight adversary, it might not be the case that $Q_{1/3}(\sigma) = \Theta(\mathrm{Madv}(\sigma))$, as the multiplicative adversary might be larger than the ϵ-error query complexity.

There is a natural extension of the multiplicative adversary that does act as a bound to ϵ-error quantum query complexity.

Definition 4.8. Let σ be a Gram matrix and $\epsilon > 0$. Then define

$$\mathrm{Madv}_\epsilon(\sigma) = \min_\rho \mathrm{Madv}(\rho)$$

where ρ ranges over all Gram matrices with $F_H(\rho, \sigma) \geq 1 - \epsilon$.

By Proposition 3.10, we have that $Q^c_\epsilon(\sigma) \geq \mathrm{Madv}_\epsilon(\sigma)$. Thus, while the multiplicative adversary does not bound ϵ-error query complexity directly, we can still use it to get interesting bounds for the ϵ-error case.

4.4 Comparing adversary methods

Each of the three adversary methods above has advantages and disadvantages. While the negative weight adversary is always stronger than the positive weight adversary, and the multiplicative adversary is stronger than both, there are still good reasons to use the weaker adversary methods.

The positive weight adversary, in particular the form of the weighted adversary of Definition 4.3, is relatively easy to apply to real world query problems. In many cases, one can assume the weights are 0 or 1 and that only adjacent inputs have nonzero weights. The objective function is then very easy to evaluate for a given weight scheme, making it very easy to show basic lower bounds. While this doesn't work for all query problems, it works for enough that the positive weight adversary is still the most popular technique for proving real-world adversary bounds. However, the positive weight adversary is weaker than the negative weight adversary and isn't mathematically simpler, so it has no advantages over the negative weight adversary when it comes to theory work.

The negative weight adversary is tight for constant-error query complexity. This makes it the method of choice for bounding that quantity. The spectral norm is well-understood, which makes it relatively easy to work with theoretically as well. While this may seem like it makes the negative weight adversary ideal for every case, there are some applications where the negative weight adversary isn't perfect. The negative weight adversary is not tight for zero-error query complexity, or for cases where the error ϵ approaches 0 or $\frac{1}{2}$ in the limit. Thus the negative weight adversary is not useful for analyzing settings with a very large chance of error, including the strong direct product theorem.

The multiplicative adversary is the strongest adversary method, but it is also the hardest to use. Thus, the multiplicative adversary is only used in the non-constant error regimes where the negative weight adversary is not tight. For example, the multiplicative adversary was used to prove the strong direct product theorem for quantum query complexity [LR12]. However, the multiplicative adversary is very difficult to use. It is challenging to compute numerically (as it doesn't take the form of a semi-definite program), it doesn't connect in an obvious way to any well-known quantities in quantum information, and there is little existing work to suggest good forms for the adversary matrix. It is not known whether the multiplicative adversary method is tight for any error regime.

In Chapter 6, I present a new formula for the multiplicative adversary that attempts to fix the problems listed above. I express the bound in terms of the max-relative entropy, which has been studied by a good amount of prior work in quantum information. I simplify the complex optimization problem by removing one of the variables, leaving only a maximization over Γ. Finally, in Theorem 6.4, I give a simple construction for Γ from any negative weight adversary matrix, which could give intuition for what kinds of adversary matrices are effective in optimizing the multiplicative adversary.

Chapter 5

Generalized Adversary Method

The lower bound proofs of the negative weight adversary and of the multiplicative adversary both make use of a construct called a progress function, which represents a kind of distance between two Gram matrices. The proofs rely on showing that the distance between the initial and final Gram matrices is very large, while performing a single query can only change the distance by a small amount, thus the number of queries must be very large.

The core difference between the negative weight adversary and the multiplicative adversary is the use of the progress function. In [Špa08], the progress function is always defined as $W^t = \langle \Gamma, \rho_I^t \rangle$ where Γ is the adversary matrix and ρ is the Gram matrix after t steps. The paper uses this adversary differently in each method: the negative weight adversary bounds the difference $W^t - W^{t+1}$, while the multiplicative adversary bounds the ratio W^t/W^{t+1}. This approach is extended in [AMRR11], which uses the same definition of progress function, but adds more conditions on the valid adversary matrices for the negative weight and multiplicative adversary methods.

Since changing the analysis of the progress function slightly from the negative weight adversary yields the multiplicative adversary, one can imagine that another change could yield a new adversary method, possibly one even stronger than the multiplicative adversary. In order to search for new ways to analyze progress functions, it is helpful to generalize away the specific choice of progress function above, and to keep only the essential conditions needed to complete the proofs. Note that bounding the ratio W^t/W^{t+1} is equivalent to bounding the difference $\log W^t - \log W^{t+1}$. In general, instead of analyzing different ways to measure the change in the progress function, it is simpler to vary the progress function and to always consider the additive distance.

5.1 Progress functions

I generalize the notion of a progress function to any asymmetric distance function on the set of positive matrices. Formally, I define a progress function as follows.

Definition 5.1 (progress function)**.** A progress function is a function $d : \mathrm{Pos}(\mathcal{H}) \times \mathrm{Pos}(\mathcal{H}) \to \mathbb{R}_{\geq 0}$ that represents a distance between matrices: $d(X \to Y)$ represents the distance starting from X and going to Y. We require that d satisfies the following properties for all $X, Y, Z \in \mathrm{Pos}(\mathcal{H})$:

1. Triangle: $d(X \to Z) \leq d(X \to Y) + d(Y \to Z)$
2. Mixing: $d(\sum_i X_i \circ Z_i \to \sum_i Y_i \circ Z_i) \leq \max_i d(X_i \to Y_i)$, where $\sum_i Z_i$ is a Gram matrix

The triangle property is fairly intuitive: if we want the progress function to represent a distance between positive matrices, then it should satisfy the basic property of a distance. The mixing property is less intuitive, but it is extremely useful when combined with Lemma 5.2. Similarly, in many of the proofs below, it is straightforward to show the triangle property, but more complex to show the mixing property. These suggest that the mixing property may not be the fundamentally correct property to use in the definition of progress functions. I leave the question of finding replacement properties as a topic for future work.

I now move onto the main result, a proof that progress functions give us a lower bound on quantum query complexity. Before I give that proof, I first show the following technical lemma. This lemma is not specific to progress functions, but it is extremely helpful for analyzing query algorithms.

Lemma 5.2. *Let A be the Gram matrix of the set of states $\{|\psi_x\rangle\}_x$, and let B be the Gram matrix of the set of states $\{O_x |\psi_x\rangle\}_x$, where O_x is the query oracle. Then there exists a set of positive matrices $\{Z_i\}$ such that $A = \sum_i J \circ Z_i$ and $B = \sum_i S_i \circ Z_i$.*

Proof. Let $i \in [n]$ be an index into the bit string x, let $|i\rangle$ be the corresponding basis state of $\mathcal{H}_{\text{query}}$, and let $|\psi_{x,i}\rangle$ denote the projection $\langle i|\psi_x\rangle$. Note that $|\psi_x\rangle = \sum_i |\psi_{x,i}\rangle$. Let Z_i be the matrix given by $\langle x|Z_i|y\rangle = \langle \psi_{x,i}|\psi_{y,i}\rangle$. Z_i is almost a Gram matrix (except that each $|\psi_{x,i}\rangle$ might not be a unit vector), so it is positive. Now, we need to check each of the sum conditions.

To show the first condition, I show that $\langle x|A|y\rangle = \langle x| \sum_i Z_i |y\rangle$. We can compute

$$\langle x|A|y\rangle = \langle \psi_x|\psi_y\rangle = \sum_i \sum_j \langle \psi_{x,i}|\psi_{y,j}\rangle$$

Since $|i\rangle$ and $|j\rangle$ are orthogonal when $i \neq j$, we know that $|\psi_{x,i}\rangle$ and $|\psi_{y,j}\rangle$ are orthogonal when $i \neq j$, so we can replace the double sum with a single sum to get

$$\langle x|A|y\rangle = \sum_i \langle \psi_{x,i}|\psi_{y,i}\rangle = \sum_i \langle x|Z_i|y\rangle = \langle x| \sum_i Z_i|y\rangle$$

which completes the proof of the first condition.

To show the second condition, I show that $\langle x|B|y\rangle = \langle x| \sum_i S_i \circ Z_i|y\rangle$. We can compute

$$\langle x|B|y\rangle = \langle O_x\psi_x|O_x\psi_y\rangle = \sum_i \sum_j \langle O_x\psi_{x,i}|O_x\psi_{y,j}\rangle$$

Note that O_x flips the sign when $x_i = 1$ and does nothing otherwise, so we have $O_x |\psi_{x,i}\rangle = (-1)^{x_i} |\psi_{x,i}\rangle$. Then by a similar argument as above, we have

$$\langle x|B|y\rangle = \sum_i (-1)^{x_i}(-1)^{y_i} \langle \psi_{x,i}|\psi_{y,i}\rangle = \sum_i (-1)^{x_i}(-1)^{y_i} \langle x|Z_i|y\rangle$$

Note that $(-1)^{x_i}(-1)^{y_i} = \langle x|S_i|y\rangle$, so the above equation simplifies to

$$\langle x|B|y\rangle = \sum_i \langle x|S_i|y\rangle \langle x|Z_i|y\rangle = \sum_i \langle x|S_i \circ Z_i|y\rangle = \langle x| \sum_i S_i \circ Z_i|y\rangle$$

which completes the proof of the second condition. □

With this lemma, it is straightforward to prove that any progress function gives us a lower bound on quantum query complexity.

Theorem 5.3. *Let σ_1, σ_2 be Gram matrices, and let d be a progress function. Then we have*

$$Q^c(\sigma_1, \sigma_2) \geq \frac{d(\sigma_1 \to \sigma_2)}{\max_i d(J \to S_i)}$$

In the case of a classical function f with Gram matrix F, the above result simplifies to

$$Q^c(f) \geq \frac{d(J \to F)}{\max_i d(J \to S_i)}$$

Proof. Assume that $Q^c(\sigma_1, \sigma_2) = T$, so we can find a query algorithm that runs for exactly T steps. Recall that this query algorithm takes the form $U_T O_x U_{T-1} ... U_1 O_x U_0$. If the input

bit string is x, let $|\psi_x^t\rangle$ denote the state of the system immediately after we run the unitary U_t. Let $\sigma^{(t)}$ denote the Gram matrix of $\{|\psi_x^t\rangle\}_x$.

Since the Gram matrix of a set of vectors is unchanged by applying a unitary to each vector, we know that the Gram matrix of $\{O_x |\psi_x^t\rangle\}_x$ is exactly $\sigma^{(t+1)}$. Then by Lemma 5.2, we can find a set $\{Z_i\}$ such that $\sigma^{(t)} = \sum_i J \circ Z_i$ and $\sigma^{(t+1)} = \sum_i S_i \circ Z_i$. Since $\sum_i Z_i = \sigma^{(t)}$ is a Gram matrix, we know by the mixing property of d that $d(\sigma^{(t)} \to \sigma^{(t+1)}) \leq \max_i d(J \to S_i)$.

Again, since Gram matrices are unchanged by unitaries, we know that $\sigma^{(0)} = \sigma_1$ and $\sigma^{(T)} = \sigma_2$. Then by the triangle property of d, we know that

$$d(\sigma_1 \to \sigma_2) = d(\sigma^{(0)} \to \sigma^{(T)}) \leq \sum_t d(\sigma^{(t)} \to \sigma^{(t+1)})$$

Combining this with the above result gives

$$d(\sigma_1 \to \sigma_2) \leq \sum_t \max_i d(J \to S_i) = T \cdot \max_i d(J \to S_i)$$

Rearranging terms, we have

$$\frac{d(\sigma_1 \to \sigma_2)}{\max_i d(J \to S_i)} \leq T = Q^c(\sigma_1, \sigma_2)$$

which concludes the proof. □

The above theorem gives us a huge set of lower bounds for quantum query complexity: any progress function gives us a new bound, so we are only limited by the number of progress functions we can invent. For the remainder of this chapter, I construct examples of progress functions.

5.2 Adversary matrices

The reader might notice that the bound on quantum query complexity above does not include any reference to an adversary matrix Γ, so the progress function does not appear to give us a new adversary method. In the following theorem, I show that adversary matrices let us transform each progress function info an infinite family of progress functions.

Theorem 5.4. *Let d be a progress function and Γ be any positive matrix. Then the function $d'(X \to Y) = d(\Gamma \circ X \to \Gamma \circ Y)$ is also a progress function.*

Proof. First, note that d' is well-defined: since the Hadamard product of two positive matrices is always positive, we know that $\Gamma \circ X$ and $\Gamma \circ Y$ are always in the domain of d. I now verify d' satisfies each of the conditions of progress functions.

1. Triangle: $d'(X \to Z) \leq d'(X \to Y) + d'(Y \to Z)$. This is equivalent to $d(\Gamma \circ X \to \Gamma \circ Z) \leq d(\Gamma \circ X \to \Gamma \circ Y) + d(\Gamma \circ Y \to \Gamma \circ Z)$, which follows directly from the triangle property of d.
2. Mixing: $d'(\sum_i X_i \circ Z_i \to \sum_i Y_i \circ Z_i) \leq \max_i d'(X_i \to Y_i)$, where $\sum_i Z_i$ is a Gram matrix. This is equivalent to $d(\sum_i \Gamma \circ X_i \circ Z_i \to \sum_i \Gamma \circ Y_i \circ Z_i) \leq \max_i d(\Gamma \circ X_i \to \Gamma \circ Y_i)$, which follows directly from the mixing property of d.

Therefore, d' is a progress function. □

Corollary 5.5. *Let σ_1, σ_2 be Gram matrices, and let d be a progress function. Then we have*

$$Q^c(\sigma_1, \sigma_2) \geq \sup_{\Gamma \in \mathrm{Pos}(\mathcal{H})} \frac{d(\Gamma \circ \sigma_1 \to \Gamma \circ \sigma_2)}{\max_i d(\Gamma \to \Gamma \circ S_i)}$$

In the case of a classical function f with Gram matrix F, the above result simplifies to

$$Q^c(f) \geq \sup_{\Gamma \in \mathrm{Pos}(\mathcal{H})} \frac{d(\Gamma \to \Gamma \circ F)}{\max_i d(\Gamma \to \Gamma \circ S_i)}$$

Proof. Combine Theorem 5.3 and Theorem 5.4. □

Thus each progress function does give us an adversary method, as we are maximizing over a set of adversary matrices. Now, all we need to do is find examples of progress functions, and each example will give us a new adversary method.

5.3 Additive adversary method

Theorem 5.6. *The function $d(X \to Y) = \|X - Y\|$ is a progress function.*

Proof. I show that d satisfies each of the properties of a progress function.

1. Triangle: This follows immediately from the triangle property of the spectral norm.
2. Mixing: We want to show that $\|\sum_i (X_i - Y_i) \circ Z_i\| \leq \max_i \|X_i - Y_i\|$. This follows from the following chain of equations:

$$\begin{aligned}
\left\|\sum_i (X_i - Y_i) \circ Z_i\right\| &= \sup_v \langle vv^*, \sum_i (X_i - Y_i) \circ Z_i\rangle \\
&= \sup_v \sum_i \langle vv^* \circ Z_i, X_i - Y_i\rangle \\
&\leq \sup_v \sum_i \|vv^* \circ Z_i\|_{tr} \|X_i - Y_i\| \\
&\leq \max_i \|X_i - Y_i\| \sup_v \sum_i \|vv^* \circ Z_i\|_{tr} \\
&= \max_i \|X_i - Y_i\|
\end{aligned}$$

For the last step, I claim that $\sup_v \sum_i \|vv^* \circ Z_i\|_{tr} = 1$. To prove this, first note that $vv^* \circ Z_i$ is a Hadamard product of positive matrices, so it is positive, so we can replace the trace norm with a trace. Then we can move the sum inside the trace to get $\sup_v \operatorname{Tr}(vv^* \circ \sum_i Z_i)$. Finally, since $\sum_i Z_i$ is a Gram matrix, taking a Hadamard product by it doesn't affect the trace, so we are left with $\sup_v \operatorname{Tr}(vv^*) = 1$. □

Since the spectral norm is a progress function, it gives us an adversary method:

Definition 5.7 (additive adversary)**.** For any Gram matrices σ_1, σ_2, define

$$\operatorname{Aadv}(\sigma_1, \sigma_2) = \sup_{\Gamma \in \operatorname{Pos}(\mathcal{H})} \frac{\|\Gamma \circ (\sigma_1 - \sigma_2)\|}{\max_i \|\Gamma \circ (J - S_i)\|}$$

To simplify the usual case of quantum state generation, also define $\operatorname{Aadv}(\sigma) = \operatorname{Aadv}(J, \sigma)$.

It turns out that the requirement that the adversary matrix Γ be positive is unimportant. By Lemma A.1, we can equivalently define Aadv as maximizing Γ over all Hermitian matrices or over all real symmetric matrices.

We know by Corollary 5.5 that $Q^c(\sigma_1, \sigma_2) \geq \operatorname{Aadv}(\sigma_1, \sigma_2)$, so we have a new lower bound for quantum query complexity. I claim this additive adversary method is equivalent to the negative weight adversary method of Definition 4.5.

Theorem 5.8. *For any classical function f with Gram matrix F, we have* $\operatorname{Aadv}(F) = \Theta(\operatorname{Adv}^{\pm}(f))$. *More precisely, we have* $\operatorname{Aadv}(F) \leq \operatorname{Adv}^{\pm}(f) \leq 2 \operatorname{Aadv}(F)$.

Proof. I first claim that any solution to $\operatorname{Aadv}(F)$ yields a solution to $\operatorname{Adv}^{\pm}(f)$. By Lemma A.1, let Γ be a real symmetric matrix that maximizes $\operatorname{Aadv}(F)$, so that $\operatorname{Aadv}(F) = \frac{\|\Gamma \circ (J-F)\|}{\max_i \|\Gamma \circ (J-S_i)\|}$. Let $\Gamma' = \Gamma \circ (J - F)$.

I claim that Γ' is a valid solution to $\mathrm{Adv}^{\pm}(f)$. To see this, note that Γ' is real symmetric by definition, and that $\Gamma' \circ F = \Gamma \circ (F - F \circ F) = \Gamma \circ (F - F) = 0$. Also recall that $D_i = (J - S_i)/2$. Then we can bound the objective value by

$$\begin{aligned}
\mathrm{Adv}^{\pm}(f) &\geq \frac{\|\Gamma'\|}{\max_i \|\Gamma' \circ D_i\|} \\
&= \frac{2\,\|\Gamma \circ (J - F)\|}{\max_i \|\Gamma' \circ (J - S_i)\|} \\
&\geq \frac{2\,\|\Gamma \circ (J - F)\|}{\max_i \gamma_2(J - F)\,\|\Gamma \circ (J - S_i)\|} \\
&\geq \frac{\|\Gamma \circ (J - F)\|}{\max_i \|\Gamma \circ (J - S_i)\|} \\
&= \mathrm{Aadv}(J, F)
\end{aligned}$$

Next, I claim that any solution to $\mathrm{Adv}^{\pm}(f)$ yields a solution to $\mathrm{Aadv}(F)$. Let Γ maximize $\mathrm{Adv}^{\pm}(f)$, so that $\mathrm{Adv}^{\pm}(f) = \frac{\|\Gamma\|}{\max_i \|\Gamma \circ D_i\|}$ and $\Gamma \circ F = 0$. This implies that $\Gamma \circ (J - F) = \Gamma \circ J = \Gamma$.

Consider Γ as a solution to $\mathrm{Aadv}(F)$. We can bound the objective value by

$$\begin{aligned}
\mathrm{Aadv}(F) &\geq \frac{\|\Gamma \circ (J - F)\|}{\max_i \|\Gamma \circ (J - S_i)\|} \\
&= \frac{\|\Gamma \circ (J - F)\|}{2 \max_i \|\Gamma \circ D_i\|} \\
&= \frac{\|\Gamma\|}{2 \max_i \|\Gamma \circ D_i\|} \\
&= \frac{\mathrm{Adv}^{\pm}(f)}{2}
\end{aligned}$$

Combining both inequalities completes the theorem. □

Chapter 6

Multiplicative Adversary Method

The previous chapter gives us a simple framework for constructing new adversary methods. I use this framework to present a new formulation of the multiplicative adversary method that is both simpler and easier to work with than the existing formulations.

To create a new adversary method, we need to find a new progress function. The core idea of this chapter is that $\mathrm{D}_{\max}$ is a progress function, and that the adversary method it gives us is exactly the multiplicative adversary method.

6.1 Redefining the multiplicative adversary

Theorem 6.1. *The function $d(X \to Y) = \mathrm{D}_{\max}(Y\|X)$ is a progress function.*

Proof. I show that d satisfies each of the properties of a progress function.

1. Triangle: This follows immediately from the triangle property of $\mathrm{D}_{\max}$.
2. Mixing: By quasiconvexity of $\mathrm{D}_{\max}$ (Proposition 2.13), we have

$$d(\sum_i X_i \circ Z_i \to \sum_i Y_i \circ Z_i) = \mathrm{D}_{\max}(\sum_i Y_i \circ Z_i \| \sum_i X_i \circ Z_i) \leq \max_i \mathrm{D}_{\max}(Y_i \circ Z_i \| X_i \circ Z_i)$$

Then, Hadamard product only decreases $\mathrm{D}_{\max}$, so we have

$$\max_i \mathrm{D}_{\max}(Y_i \circ Z_i \| X_i \circ Z_i) \leq \max_i \mathrm{D}_{\max}(Y_i \| X_i) = \max_i d(X_i \to Y_i)$$

Combining these chains of equations finishes the proof. □

By Corollary 5.5, this immediately gives us a new adversary method:

$$Q^c(\sigma_1, \sigma_2) \geq \mathrm{Madv}(\sigma_1, \sigma_2) = \sup_{\Gamma \in \mathrm{Pos}(\mathcal{H})} \frac{\mathrm{D}_{\max}(\Gamma \circ \sigma_2 \| \Gamma \circ \sigma_1)}{\max_i \mathrm{D}_{\max}(\Gamma \circ S_i \| \Gamma)}$$

I claim that this adversary method is exactly equal to the multiplicative adversary method of Definition 4.7.

Theorem 6.2. *For any Gram matrix σ, we have* $\mathrm{Madv}(J, \sigma) = \mathrm{Madv}(\sigma)$.

Proof. I first prove that any solution to $\mathrm{Madv}(J, \sigma)$ yields a solution to $\mathrm{Madv}(\sigma)$. Let Γ maximize $\mathrm{Madv}(J, \sigma)$, so that $\mathrm{Madv}(J, \sigma) = \frac{\mathrm{D}_{\max}(\Gamma \circ \sigma \| \Gamma)}{\max_i \mathrm{D}_{\max}(\Gamma \circ S_i \| \Gamma)}$. Let $Z \in \mathrm{Pos}(\mathcal{H})$ maximize $\frac{\langle \Gamma \circ \sigma, Z \rangle}{\langle \Gamma, Z \rangle}$, so that $\mathrm{D}_{\max}(\Gamma \circ \sigma \| \Gamma) = \log \frac{\langle \Gamma \circ \sigma, Z \rangle}{\langle \Gamma, Z \rangle}$. Finally, define $c = \max_i R(\Gamma \circ S_i \| \Gamma)$, and let $\Gamma' = \Gamma \circ Z$.

Assume without loss of generality that Z is scaled such that $\mathrm{Tr}(\Gamma' J) = 1$. This implies that $\langle \Gamma, Z \rangle = 1$, so $\mathrm{D}_{\max}(\Gamma \circ \sigma \| \Gamma) = \log \langle \Gamma \circ \sigma, Z \rangle$.

I claim that c, Γ' is a valid solution to $\mathrm{Madv}(\sigma)$. We have $c > 1$ by a property of R, and Γ' is positive since Γ and Z are both positive. We clearly have $\mathrm{Tr}(\Gamma' J) = 1$, and we also know that $\Gamma' \circ S_i \leq R(\Gamma' \circ S_i \| \Gamma') \cdot \Gamma' \leq R(\Gamma \circ S_i \| \Gamma) \cdot \Gamma' \leq c\Gamma'$, so both conditions hold, and the solution is valid.

Then we can bound the objective value of $\mathrm{Madv}(\sigma)$ as

$$\begin{aligned}
\mathrm{Madv}(\sigma) &\geq \frac{\log \mathrm{Tr}(\Gamma' \sigma)}{\log c} \\
&= \frac{\log \langle \Gamma \circ Z, \sigma \rangle}{\max_i \mathrm{D}_{\max}(\Gamma \circ S_i \| \Gamma)} \\
&= \frac{\mathrm{D}_{\max}(\Gamma \circ \sigma \| \Gamma)}{\max_i \mathrm{D}_{\max}(\Gamma \circ S_i \| \Gamma)} \\
&= \mathrm{Madv}(J, \sigma)
\end{aligned}$$

Next, I prove that any solution to $\mathrm{Madv}(\sigma)$ yields a solution to $\mathrm{Madv}(J, \sigma)$. Let $c > 1$, $\Gamma \in \mathrm{Pos}(\mathcal{H})$ maximize $\mathrm{Madv}(\sigma)$, so that $\mathrm{Madv}(\sigma) = \frac{\log \mathrm{Tr}(\Gamma \sigma)}{\log c}$ and $\mathrm{Tr}(\Gamma J) = 1$ and $\Gamma \circ S_i \leq c\Gamma$ for all i.

Consider Γ as a solution to $\mathrm{Madv}(J, \sigma)$. Note that $\Gamma \circ S_i \leq c\Gamma$ implies that $R(\Gamma \circ S_i \| \Gamma) \leq c$. Then we can bound the objective value of $\mathrm{Madv}(J, \sigma)$ as

$$\begin{aligned}
\mathrm{Madv}(J, \sigma) &\geq \frac{\mathrm{D}_{\max}(\Gamma \circ \sigma \| \Gamma)}{\max_i \mathrm{D}_{\max}(\Gamma \circ S_i \| \Gamma)} \\
&\geq \frac{\mathrm{D}_{\max}(\Gamma \circ \sigma \| \Gamma)}{\log c} \\
&= \frac{\max_{Z \in \mathrm{Pos}(\mathcal{H})} \log \frac{\langle \Gamma \circ \sigma, Z \rangle}{\langle \Gamma, Z \rangle}}{\log c} \\
&\geq \frac{\log \frac{\langle \Gamma \circ \sigma, J \rangle}{\langle \Gamma, J \rangle}}{\log c} \\
&= \frac{\log \mathrm{Tr}(\Gamma \sigma)}{\log c} \\
&= \mathrm{Madv}(\sigma)
\end{aligned}$$

Since we have $\mathrm{Madv}(\sigma) \geq \mathrm{Madv}(J, \sigma)$ and $\mathrm{Madv}(J, \sigma) \geq \mathrm{Madv}(\sigma)$, I conclude that $\mathrm{Madv}(J, \sigma) = \mathrm{Madv}(\sigma)$. □

For the remainder of this chapter and the next chapter, I use this new definition of the multiplicative adversary to reprove known results about the method. I claim these new proofs are simpler than the original proofs, suggesting this new definition is the right one to use in future work.

6.2 Madv is greater than Aadv

In this section, I prove that for any σ_1, σ_2, we have $\mathrm{Madv}(\sigma_1, \sigma_2) \geq \mathrm{Aadv}(\sigma_1, \sigma_2)$. This proof is based on the proof of Lemma 19 in [AMRR11].

I first prove a lemma that relates the spectral norm and max-relative entropy.

Lemma 6.3. *Fix some matrix $\Gamma \in \mathrm{Pos}(\mathcal{H})$. Then define a function $f : \mathbb{R} \times \mathrm{Pos}(\mathcal{H}) \times \mathrm{Pos}(\mathcal{H}) \to \mathbb{R}$ by*

$$f(\alpha, \sigma_1, \sigma_2) = \mathrm{D}_{\max}(\exp(\alpha\Gamma) \circ \sigma_2 \| \exp(\alpha\Gamma) \circ \sigma_1)$$

Then for any pair of Gram matrices σ_1, σ_2, we have

$$\|\Gamma \circ (\sigma_1 - \sigma_2)\| = \max \left\{ \lim_{\alpha \to 0^+} \frac{f(\alpha, \sigma_1, \sigma_2)}{|\alpha|}, \lim_{\alpha \to 0^-} \frac{f(\alpha, \sigma_1, \sigma_2)}{|\alpha|} \right\}$$

Proof. First, note that f is well-defined because exp always outputs a positive matrix for any input, and the Hadamard product of two positive matrices is positive, so both arguments to $\mathrm{D}_{\max}$ must be positive. Next, note that for any Hermitian matrix X, we have $\|X\| = \max\{\lambda_{max}(X), \lambda_{max}(-X)\}$.

I start by simplifying the expression for $f(\alpha, \sigma_1, \sigma_2)$ when α is small. Since we are taking the limit $\alpha \to 0$, we can discard all terms that are $O(\alpha^2)$ or smaller. Thus we can compute

$$\begin{aligned}
f(\alpha, \sigma_1, \sigma_2) &= \mathrm{D}_{\max}(\exp(\alpha\Gamma) \circ \sigma_2 \| \exp(\alpha\Gamma) \circ \sigma_1) \\
&= \log \lambda_{max}((\exp(\alpha\Gamma) \circ \sigma_2)(\exp(\alpha\Gamma) \circ \sigma_1)^{-1}) \\
&\approx \log \lambda_{max}(((\mathbb{I} + \alpha\Gamma) \circ \sigma_2)((\mathbb{I} + \alpha\Gamma) \circ \sigma_1)^{-1}) \\
&= \log \lambda_{max}((\mathbb{I} + \alpha\Gamma \circ \sigma_2)(\mathbb{I} + \alpha\Gamma \circ \sigma_1)^{-1}) \\
&\approx \log \lambda_{max}((\mathbb{I} + \alpha\Gamma \circ \sigma_2)(\mathbb{I} - \alpha\Gamma \circ \sigma_1)) \\
&\approx \log \lambda_{max}(\mathbb{I} + \alpha\Gamma \circ (\sigma_2 - \sigma_1)) \\
&= \log(1 + \lambda_{max}(\alpha\Gamma \circ (\sigma_2 - \sigma_1))) \\
&\approx \lambda_{max}(\alpha\Gamma \circ (\sigma_2 - \sigma_1)) \\
&= |\alpha| \lambda_{max}(sign(\alpha) \cdot \Gamma \circ (\sigma_2 - \sigma_1))
\end{aligned}$$

Then we have

$$\begin{aligned}
&\max\left\{\lim_{\alpha\to 0^+} \frac{f(\alpha, \sigma_1, \sigma_2)}{|\alpha|}, \lim_{\alpha\to 0^-} \frac{f(\alpha, \sigma_1, \sigma_2)}{|\alpha|}\right\} \\
&= \max\left\{\lim_{\alpha\to 0^+} \lambda_{max}(sign(\alpha) \cdot \Gamma \circ (\sigma_2 - \sigma_1)), \lim_{\alpha\to 0^-} \lambda_{max}(sign(\alpha) \cdot \Gamma \circ (\sigma_2 - \sigma_1))\right\} \\
&= \max\left\{\lambda_{max}(\Gamma \circ (\sigma_2 - \sigma_1)), \lambda_{max}(-\Gamma \circ (\sigma_2 - \sigma_1))\right\} \\
&= \|\Gamma \circ (\sigma_1 - \sigma_2)\|
\end{aligned}$$

which concludes the proof. □

Theorem 6.4. *For any Gram matrices σ_1, σ_2, we have* $\mathrm{Madv}(\sigma_1, \sigma_2) \geq \mathrm{Aadv}(\sigma_1, \sigma_2)$.

Proof. Let Γ maximize $\mathrm{Aadv}(\sigma_1, \sigma_2)$. Then we have

$$\begin{aligned}
&\mathrm{Aadv}(\sigma_1, \sigma_2) \\
&= \frac{\|\Gamma \circ (\sigma_1 - \sigma_2)\|}{\max_i \|\Gamma \circ (J - S_i)\|} \\
&= \frac{\max\left\{\lim_{\alpha\to 0^+} \frac{f(\alpha,\sigma_1,\sigma_2)}{|\alpha|}, \lim_{\alpha\to 0^-} \frac{f(\alpha,\sigma_1,\sigma_2)}{|\alpha|}\right\}}{\max_i \|\Gamma \circ (J - S_i)\|} \\
&= \max\left\{\lim_{\alpha\to 0^+} \frac{1}{|\alpha|}\frac{f(\alpha,\sigma_1,\sigma_2)}{\max_i \|\Gamma \circ (J - S_i)\|}, \lim_{\alpha\to 0^-} \frac{1}{|\alpha|}\frac{f(\alpha,\sigma_1,\sigma_2)}{\max_i \|\Gamma \circ (J - S_i)\|}\right\} \\
&\leq \max\left\{\lim_{\alpha\to 0^+} \frac{f(\alpha,\sigma_1,\sigma_2)}{\max_i f(\alpha, J, S_i)}, \lim_{\alpha\to 0^-} \frac{f(\alpha,\sigma_1,\sigma_2)}{\max_i f(\alpha, J, S_i)}\right\} \\
&\leq \mathrm{Madv}(\sigma_1, \sigma_2)
\end{aligned}$$

where the last line follows from noting that $\frac{f(\alpha,\sigma_1,\sigma_2)}{\max_i f(\alpha,J,S_i)}$ is exactly the formula you get from considering $\exp(\alpha\Gamma)$ as a possible solution to $\mathrm{Madv}(\sigma_1, \sigma_2)$. □

As a side note, we can show that $d(X \to Y) = \max\{\mathrm{D}_{\max}(X\|Y), \mathrm{D}_{\max}(Y\|X)\} = \delta_\infty(X, Y)$ is a progress function, so it yields an adversary method. It turns out that this adversary method is at least as strong as $\mathrm{D}_{\max}$, and using it instead of Madv happens to eliminate the complex max cases in the above proof.

6.3 The direct sum theorem

The direct sum theorem states that the query complexity of solving a problem k times is at least k times the complexity of solving it once. In this section, I give a simple proof of this theorem for Madv.

I first prove a lemma that describes the behavior of the denominator of Madv under tensor products. This lemma will also be useful in the proof of the strong direct product theorem in the next section.

Lemma 6.5. *For any $\Gamma \in \mathrm{Pos}(\mathcal{H})$ and any positive integer k, we have*

$$\max_{j\in[kn]} \mathrm{D}_{\max}(\Gamma^{\otimes k} \circ S_j \| \Gamma^{\otimes k}) = \max_{i\in[n]} \mathrm{D}_{\max}(\Gamma \circ S_i \| \Gamma)$$

Proof. Let $j = mn + r$, where $r \in [n]$. Then we have

$$\begin{aligned}
\max_{j\in[kn]} \mathrm{D}_{\max}(\Gamma^{\otimes k} \circ S_j \| \Gamma^{\otimes k}) &= \max_{j\in[kn]} \mathrm{D}_{\max}(\Gamma^{\otimes k} \circ (J^{\otimes m} \otimes S_r \otimes J^{\otimes(k-m-1)}) \| \Gamma^{\otimes k}) \\
&= \max_{j\in[kn]} \mathrm{D}_{\max}((\Gamma \circ J)^{\otimes m} \otimes (\Gamma \circ S_r) \otimes (\Gamma \circ J)^{\otimes(k-m-1)} \| \Gamma^{\otimes k}) \\
&= \max_{j\in[kn]} \mathrm{D}_{\max}(\Gamma \circ S_r \| \Gamma) + (k-1)\,\mathrm{D}_{\max}(\Gamma \circ J \| \Gamma) \\
&= \max_{j\in[kn]} \mathrm{D}_{\max}(\Gamma \circ S_r \| \Gamma) \\
&= \max_{i\in[n]} \mathrm{D}_{\max}(\Gamma \circ S_i \| \Gamma)
\end{aligned}$$

Note that S_j is an operator on $\mathcal{H}^{\otimes k}$, while S_i and S_r are operators on $\mathcal{H}$. □

The direct sum theorem then follows trivially from the tensor property of $\mathrm{D}_{\max}$.

Theorem 6.6 (direct sum theorem)**.** *For any Gram matrices σ_1, σ_2 and any positive integer k, we have* $\mathrm{Madv}(\sigma_1^{\otimes k}, \sigma_2^{\otimes k}) \geq k \cdot \mathrm{Madv}(\sigma_1, \sigma_2)$.

Proof. Let Γ maximize $\mathrm{Madv}(\sigma_1, \sigma_2)$. Then we have

$$\begin{aligned}
\mathrm{Madv}(\sigma_1^{\otimes k}, \sigma_2^{\otimes k}) &= \max_{\Gamma' \in \mathrm{Pos}(\mathcal{H}^{\otimes k})} \frac{\mathrm{D}_{\max}(\Gamma' \circ \sigma_2^{\otimes k} \| \Gamma' \circ \sigma_1^{\otimes k})}{\max_{j\in[kn]} \mathrm{D}_{\max}(\Gamma' \circ S_j \| \Gamma')} \\
&\geq \frac{\mathrm{D}_{\max}(\Gamma^{\otimes k} \circ \sigma_2^{\otimes k} \| \Gamma^{\otimes k} \circ \sigma_1^{\otimes k})}{\max_{j\in[kn]} \mathrm{D}_{\max}(\Gamma^{\otimes k} \circ S_j \| \Gamma^{\otimes k})} \\
&= \frac{k\,\mathrm{D}_{\max}(\Gamma \circ \sigma_2 \| \Gamma \circ \sigma_1)}{\max_{j\in[kn]} \mathrm{D}_{\max}(\Gamma^{\otimes k} \circ S_j \| \Gamma^{\otimes k})} \\
&= \frac{k\,\mathrm{D}_{\max}(\Gamma \circ \sigma_2 \| \Gamma \circ \sigma_1)}{\max_{i\in[n]} \mathrm{D}_{\max}(\Gamma \circ S_i \| \Gamma)} \\
&= k \cdot \mathrm{Madv}(\sigma_1, \sigma_2)
\end{aligned}$$

If no Γ attains the maximum of $\mathrm{Madv}(\sigma_1, \sigma_2)$, then the same proof works by letting Γ approach arbitrarily close to the value of the supremum. □

I conjecture that this proof can be extended to show that for any list of query problems, the complexity of solving all the problems at once is at least equal to the sum of the complexities of each problem. Proving this would require a guarantee that we can always find a Γ that maximizes Madv with a particular value in the denominator.

Chapter 7

Strong Direct Product Theorem

In this chapter, I reprove a result known as the strong direct product theorem. This theorem states that the quantum query complexity of solving k copies of a problem is at least k times the query complexity of solving one copy of the problem. This differs from the direct sum theorem above in that it also accounts for error. The direct product theorem says that any algorithm that uses fewer than the minimum number of queries will have an exponential decay in its success probability.

The proof in this section is based on the proof of the strong direct product theorem in [LR12]. I believe that my proof is easier to follow, as while I don't change the overall proof structure, I start from a simpler formula for Madv.

7.1 Other direct product theorems

The notion of a direct product theorem is not unique to quantum query complexity. Similar direct product theorems are known for other domains within query complexity and communication complexity:

- In randomized query complexity, [Dru12] proved that any randomized algorithm that attempts to solve $f^{\otimes k}$ with $O(\gamma^3 k R_\epsilon(f))$ queries (where ϵ is a fixed constant) has success probability at most $(1/2+\gamma)^k$.
- In quantum communication complexity, [She12] proved a slightly weaker result: that solving a quantum communication problem $f^{\otimes k}$ with success probability at least $2^{-\Omega(k)}$ requires $\Omega(k)$ qubits of communication.

- Similar results have been shown for classical communication complexity [BRWY13], for multi-party classical communication complexity [BPSW06], and for multi-party quantum communication complexity [JK22].

The strong direct product theorem is only known for query complexity and approximate degree problems. The direct product theorems for other models, such as communication complexity, are usually weaker.

In this thesis, I focus only on the case of quantum query complexity, though I generalize over both classical functions and state generation problems. I hypothesize that the strong direct product theorem also holds for state conversion, though I don't prove this result.

7.2 Lemmas

Before I prove the direct product theorem, I first need to prove several lemmas:

1. Lemma 7.1 relates fidelity and the eigenvalues of the adversary
2. Lemma 7.2 gives a formula for $\mathrm{D}_{\max}$ on linear combinations
3. Lemma 7.3 gives a specific solution to Madv from Aadv

I first prove a lemma that connects fidelity and matrix inner products. This will be used to connect the Hadamard product fidelity output condition with the supremum over inner products definition of $\mathrm{D}_{\max}$.

Lemma 7.1. *If ρ, σ are mixed states and Γ is positive, then*

$$\langle \Gamma^{\otimes k}, \rho \rangle \geq F(\rho, \sigma^{\otimes k}) \left(\frac{\lambda_{min}(\Gamma)\lambda_{max}(\Gamma)}{\lambda_{min}(\Gamma) + \lambda_{max}(\Gamma) - \langle \Gamma, \sigma \rangle} \right)^k$$

Proof. Note that this lemma is not specific to quantum query complexity, so its proof is not the main focus of this thesis. As such, I start from a known related result instead of proving it from scratch. Corollary 3.13 of [LR12] states that for any $a_1 \geq a_0 > 0$, if p is a distribution for a random variable A taking values in $[a_0, a_1]$, if $E_p[A] = \bar{a}$, if q is a distribution over $(\mathbb{R}^+)^k$, and if $F(p^{\otimes k}, q) \geq \delta^k$, then

$$E_q \left[\prod_{l=1}^{k} A_l \right] \geq \left(\frac{\delta a_0 a_1}{a_0 + a_1 - \bar{a}} \right)^k$$

Note that this statement is slightly different from the one in [LR12]: the paper uses square-root fidelity, while I use normal fidelity.

This lemma follows directly from the above result. Let p be a measurement of Γ on σ and q be a measurement of $\Gamma^{\otimes k}$ on ρ. In both cases, the range of the measurement is the same as the range of the eigenvalues of Γ, so we have $a_0 = \lambda_{min}(\Gamma)$ and $a_1 = \lambda_{max}(\Gamma)$. The expected value of a measurement is just the inner product, so we have $\bar{a} = \langle \Gamma, \sigma \rangle$. Finally, we know that $F(\rho, \sigma^{\otimes k}) \leq F(p^{\otimes k}, q)$, since measuring a pair of mixed states can only increase their fidelity. Combining all of the above results completes the proof. □

Next, I prove a lemma that exactly characterizes the behavior of $\mathrm{D}_{\max}$ on particular linear inputs. This lets us compute properties of $\mathrm{D}_{\max}$ that we need for the final lemma.

Lemma 7.2. *Let ρ, X be Hermitian operators that aren't negative semi-definite. Let c_1, c_2 be real numbers with $c_1 > c_2$, and assume $\rho + c_2 X$ is positive. Then*

$$R(\rho + c_1 X \| \rho + c_2 X) = \frac{1 + c_1 R(X\|\rho)}{1 + c_2 R(X\|\rho)}$$

$$\mathrm{D}_{\max}(\rho + c_1 X \| \rho + c_2 X) = \log\left(\frac{1 + c_1 \exp(\mathrm{D}_{\max}(X\|\rho))}{1 + c_2 \exp(\mathrm{D}_{\max}(X\|\rho))}\right)$$

Proof. First, since X is not negative, we have $(c_1 - c_2)X \not\preceq 0$, so $\rho + c_1 X \not\preceq \rho + c_2 X$, so we know that $R(\rho + c_1 X \| \rho + c_2 X) \geq 1$. Second, since $\rho + c_2 X$ is positive, we know that $R(\rho + c_2 X \| \rho) = 1 + c_2 R(X\|\rho)$ is also positive. Then we can compute

$$\begin{aligned}
R(\rho + c_1 X \| \rho + c_2 X) &= \min\{\lambda \geq 1 : c_1 X + \rho \preceq \lambda(c_2 X + \rho)\} \\
&= \min\{\lambda \geq 1 : (c_1 - c_2\lambda)X \preceq (\lambda - 1)\rho\} \\
&= \min\{\lambda \geq 1 : (c_1 - c_2\lambda)R(X\|\rho) \leq \lambda - 1\} \\
&= \min\{\lambda \geq 1 : 1 + c_1 R(X\|\rho) \leq \lambda(1 + c_2 R(X\|\rho))\} \\
&= \frac{1 + c_1 R(X\|\rho)}{1 + c_2 R(X\|\rho)}
\end{aligned}$$

The other result follows immediately from the definition of R. □

In particular, note that the above lemma always applies when ρ is a Gram matrix and X is a Hermitian operator with zeros on the diagonal.

Finally, I prove a lemma that guarantees we can find a nice solution for Madv given any solution to Aadv. The adversary matrix is guaranteed to have several useful properties that simultaneously bound the value of Madv and let us apply Lemma 7.1 above. This lemma requires an unusual condition, namely that $(J - \sigma) \circ (J - \sigma) = \lambda(J - \sigma)$. This condition is always true with $\lambda = 1$ for classical functions.

Lemma 7.3. *Let σ be a Gram matrix and $\gamma > 0$. Assume that $(J-\sigma)\circ(J-\sigma) = \lambda(J-\sigma)$ for some $\lambda \in [0,2]$. Then there exists a matrix Γ_m and a unit vector v satisfying:*

1. $\lambda_{min}(\Gamma_m) = 1$
2. $\lambda_{max}(\Gamma_m) \leq 1 + 2\gamma \operatorname{Aadv}(\sigma)$
3. $\max_i \mathrm{D}_{\max}(\Gamma_m \circ S_i \| \Gamma_m) \leq \log(1+2\gamma)$
4. $v^*(\Gamma_m \circ \sigma)v = 1 + \lambda\gamma \operatorname{Aadv}(\sigma)$
5. $v^*\Gamma_m v = 1$

Proof. Let Γ maximize $\operatorname{Aadv}(\sigma)$, so that

$$\operatorname{Aadv}(\sigma) = \frac{\|\Gamma \circ (J-\sigma)\|}{\max_i \|\Gamma \circ (J - S_i)\|}$$

Assume without loss of generality that $\lambda_{max}(\Gamma \circ (J-\sigma)) = \|\Gamma \circ (J-\sigma)\| = 1$. If this equation doesn't hold, then we can simply scale Γ by the appropriate (positive or negative) real constant without changing Aadv.

To simplify the notation, I define $A = \operatorname{Aadv}(\sigma)$ and $\Gamma' = \Gamma \circ (J-\sigma)$.

Let $\Gamma_m = (1+\gamma A)\cdot \mathbb{I} - \gamma A \cdot \Gamma'$, and let v be the principal eigenvector of Γ'.

I verify each condition in order:

$$\lambda_{min}(\Gamma_m) = 1 + \gamma A - \gamma A \cdot \lambda_{max}(\Gamma') = 1 + \gamma A - \gamma A = 1$$

$$\lambda_{max}(\Gamma_m) = 1 + \gamma A - \gamma A \cdot \lambda_{min}(\Gamma') \leq 1 + \gamma A + \gamma A \|\Gamma'\| = 1 + 2\gamma A$$

$$
\begin{aligned}
\max_i \mathrm{D}_{\max}(\Gamma_m \circ S_i \| \Gamma_m) &= \max_i \mathrm{D}_{\max}(((1+\gamma A)\cdot \mathbb{I} - \gamma A \cdot \Gamma') \circ S_i \| (1+\gamma A)\cdot \mathbb{I} - \gamma A \cdot \Gamma') \\
&= \max_i \mathrm{D}_{\max}((1+\gamma A)\cdot \mathbb{I} - \gamma A \cdot \Gamma' \circ S_i \| (1+\gamma A)\cdot \mathbb{I} - \gamma A \cdot \Gamma') \\
&= \max_i \log(1 + R(\gamma A \cdot \Gamma' \circ (J - S_i) \| (1+\gamma A)\cdot \mathbb{I} - \gamma A \cdot \Gamma')) \\
&\leq \max_i \log(1 + R(\gamma A \cdot \Gamma' \circ (J - S_i) \| (1+\gamma A)\cdot \mathbb{I} - \gamma A \, \|\Gamma'\| \cdot \mathbb{I})) \\
&= \max_i \log(1 + R(\gamma A \cdot \Gamma' \circ (J - S_i) \| \mathbb{I})) \\
&= \max_i \log(1 + \lambda_{max}(\gamma A \cdot \Gamma' \circ (J - S_i))) \\
&\leq \max_i \log(1 + \gamma A \cdot \|\Gamma' \circ (J - S_i)\|) \\
&= \max_i \log\left(1 + \gamma \cdot \frac{\|\Gamma \circ (J - S_i) \circ (J - \sigma)\|}{\|\Gamma \circ (J - S_i)\|}\right) \\
&\leq \max_i \log(1 + \gamma \cdot \gamma_2(J - \sigma)) \\
&\leq \max_i \log(1 + 2\gamma)
\end{aligned}
$$

$$
\begin{aligned}
v^*(\Gamma_m \circ \sigma)v &= v^*((1+\gamma A)\cdot \mathbb{I} \circ \sigma - \gamma A \cdot \Gamma' \circ \sigma)v \\
&= (1+\gamma A)\cdot v^* \mathbb{I} v - \gamma A \cdot v^*(\Gamma' \circ \sigma)v \\
&= 1 + \gamma A - \gamma A \cdot v^*((1-\lambda)\Gamma')v \\
&= 1 + \gamma A - (1-\lambda)\gamma A \\
&= 1 + \lambda \gamma A
\end{aligned}
$$

$$
\begin{aligned}
v^* \Gamma_m v &= v^*((1+\gamma A)\cdot \mathbb{I} - \gamma A \cdot \Gamma')v \\
&= (1+\gamma A)\cdot v^* \mathbb{I} v - \gamma A \cdot v^* \Gamma' v \\
&= 1 + \gamma A - \gamma A \\
&= 1
\end{aligned}
$$

□

7.3 Main result

I now prove the main result of this section, a strong direct product theorem for quantum query complexity.

Theorem 7.4. *Let σ be a Gram matrix, k be a positive integer, $\delta \in [0,1]$, and $\beta > 0$. Assume that $(J-\sigma)\circ(J-\sigma) = \lambda(J-\sigma)$. Then*

$$\mathrm{Madv}_{1-\delta^k}(\sigma^{\otimes k}) \geq \frac{k \log(\delta \frac{1+2\beta}{1+(2-\lambda)\beta})}{2\beta} \mathrm{Aadv}(\sigma)$$

For instance, if the condition holds for $\lambda = 1$, we can pick $\beta = 1$ to get

$$\mathrm{Madv}_{1-\delta^k}(\sigma^{\otimes k}) \geq \frac{k \log(\delta \cdot 3/2)}{2} \mathrm{Aadv}(\sigma)$$

Proof. For an arbitrary $\gamma > 0$, let Γ_m and v be given by Lemma 7.3. Let ρ vary over all Gram matrices that satisfy $F_H(\rho, \sigma^{\otimes k}) \geq \delta^k$. Then we have

$$\begin{aligned}
\mathrm{Madv}_{1-\delta^k}(\sigma^{\otimes k}) &= \min_\rho \mathrm{Madv}(\rho) \\
&= \min_\rho \max_{\Gamma \in \mathrm{Pos}(\mathcal{H}^{\otimes k})} \frac{\mathrm{D}_{\max}(\Gamma \circ \rho \| \Gamma)}{\max_{j\in[kn]} \mathrm{D}_{\max}(\Gamma \circ S_j \| \Gamma)} \\
&\geq \min_\rho \frac{\mathrm{D}_{\max}(\Gamma_m^{\otimes k} \circ \rho \| \Gamma_m^{\otimes k})}{\max_{j\in[kn]} \mathrm{D}_{\max}(\Gamma_m^{\otimes k} \circ S_j \| \Gamma_m^{\otimes k})} \\
&= \min_\rho \frac{\mathrm{D}_{\max}(\Gamma_m^{\otimes k} \circ \rho \| \Gamma_m^{\otimes k})}{\max_{i\in[n]} \mathrm{D}_{\max}(\Gamma_m \circ S_i \| \Gamma_m)} \\
&\geq \frac{1}{\log(1+2\gamma)} \min_\rho \mathrm{D}_{\max}(\Gamma_m^{\otimes k} \circ \rho \| \Gamma_m^{\otimes k})
\end{aligned}$$

Next, I use the supremum definition of $\mathrm{D}_{\max}(\Gamma_m^{\otimes k} \circ \rho \| \Gamma_m^{\otimes k})$ and Lemma 7.1 to get

$$\begin{aligned}
&\mathrm{D}_{\max}(\Gamma_m^{\otimes k} \circ \rho \| \Gamma_m^{\otimes k}) \\
&= \sup_{A \in D(\mathcal{H}^{\otimes k})} \log \frac{\langle A, \Gamma_m^{\otimes k} \circ \rho \rangle}{\langle A, \Gamma_m^{\otimes k} \rangle} \\
&\geq \log \frac{\langle (vv^*)^{\otimes k}, \Gamma_m^{\otimes k} \circ \rho \rangle}{\langle (vv^*)^{\otimes k}, \Gamma_m^{\otimes k} \rangle} \\
&= \log \langle \Gamma_m^{\otimes k}, (vv^*)^{\otimes k} \circ \rho \rangle \\
&\geq \log F((vv^*)^{\otimes k} \circ \rho, (vv^*)^{\otimes k} \circ \sigma^{\otimes k}) \left(\frac{\lambda_{min}(\Gamma_m)\lambda_{max}(\Gamma_m)}{\lambda_{min}(\Gamma_m) + \lambda_{max}(\Gamma_m) - \langle \Gamma_m, (vv^*) \circ \sigma \rangle} \right)^k
\end{aligned}$$

Since ρ varies over all Gram matrices with $F_H(\rho, \sigma^{\otimes k}) \geq \delta^k$, we know that $\min_\rho F((vv^*)^{\otimes k} \circ \rho, (vv^*)^{\otimes k} \circ \sigma^{\otimes k}) \geq \delta^k$. Combining this with the first chain of equations yields

$$\begin{aligned}
\mathrm{Madv}_{1-\delta^k}(\sigma^{\otimes k}) &\geq \frac{1}{\log(1+2\gamma)} \log \delta^k \left(\frac{\lambda_{min}(\Gamma_m)\lambda_{max}(\Gamma_m)}{\lambda_{min}(\Gamma_m) + \lambda_{max}(\Gamma_m) - \langle \Gamma_m, (vv^*) \circ \sigma \rangle} \right)^k \\
&\geq \frac{k}{\log(1+2\gamma)} \log \left(\delta \frac{1 + 2\gamma \operatorname{Aadv}(\sigma)}{1 + 2\gamma \operatorname{Aadv}(\sigma) - \lambda\gamma \operatorname{Aadv}(\sigma)} \right) \\
&= \frac{k \log \left(\delta \frac{1+2\gamma \operatorname{Aadv}(\sigma)}{1+\gamma \operatorname{Aadv}(\sigma)(2-\lambda)} \right)}{\log(1+2\gamma)}
\end{aligned}$$

Then substitute $\gamma = \beta / \operatorname{Aadv}(\sigma)$ to get

$$\begin{aligned}
\mathrm{Madv}_{1-\delta^k}(\sigma) &\geq \frac{k \log(\delta \frac{1+2\beta}{1+(2-\lambda)\beta})}{\log(1 + 2\beta/\operatorname{Aadv}(\sigma))} \\
&\geq \frac{\operatorname{Aadv}(\sigma)}{2\beta} k \log(\delta \frac{1+2\beta}{1+(2-\lambda)\beta}) \\
&= \frac{k \log(\delta \frac{1+2\beta}{1+(2-\lambda)\beta})}{2\beta} \operatorname{Aadv}(\sigma)
\end{aligned}$$

□

Corollary 7.5. *Let f be a classical function with Gram matrix F, k be a positive integer, and $\delta \in (2/3, 1]$. Then*

$$Q^c_{1-\delta^k}(f^{\otimes k}) \geq \frac{k \log(\delta \cdot 3/2)}{2} \operatorname{Aadv}(F)$$

$$Q_{1-\delta^{k/2}}(f^{\otimes k}) \geq \frac{k \log(\delta \cdot 3/2)}{4} \operatorname{Aadv}(F)$$

Proof. Note that F is a 0-1 matrix, so $(J - F) \circ (J - F) = (J - F)$. Therefore we can apply Theorem 7.4 with $\lambda = \beta = 1$ to get the first result. The second result follows from the fact that $Q^c_\epsilon(F) \leq 2Q_{1-\sqrt{1-\epsilon}}(F)$, which is proved as Claim 2.5 in [LR12]. □

Note that Corollary 7.5 is exactly equivalent to Theorem 4.2 of [LR12].

Chapter 8

Conclusion and Future Work

I presented a generalized adversary method that unifies and simplifies the proofs of both the negative weight adversary and multiplicative adversary. This led to a new formula for the multiplicative adversary in terms of max-relative entropy. I used this new formula to reprove existing results on the multiplicative adversary, including the strong direct product theorem for quantum query complexity.

In the near term, the most useful of these results is likely the new formula for the multiplicative adversary. Max-relative entropy is much easier to work with than the complex optimization programs used to define the multiplicative adversary thus far. This could lead to the multiplicative adversary being used to prove lower bounds on a much wider variety of query problems.

There are many open questions regarding this new generalized method. The most important such question is whether we can use this generalization to construct a new adversary method that lets us prove new lower bounds for query problems. Constructing a new adversary method requires two steps. First, we need to find a new way to measure the distance between two Gram matrices, and we need to prove this measure satisfies the definition of a progress function. Second, we need to show that the new adversary method is stronger or simpler than the existing methods for some real query problem.

8.1 Other progress functions

There are two natural classes of functions that are candidates to be progress functions. First, any matrix norm $\|X\|$ naturally corresponds to a potential progress function d by

letting $d(X \to Y) = \|X - Y\|$. The negative weight adversary is constructed in this way from the spectral norm. Second, any way to measure the divergence or entropy between pairs of matrices might give another progress function, like how the multiplicative adversary comes from the max-relative entropy.

I have a few initial results towards finding useful new progress functions. I wrote code to evaluate a variety of matrix norms and divergences, and I found numerical counterexamples that suggest that the following quantities **do not** yield progress functions:

- The Schatten p-norm for $p < \infty$, including the trace norm and the Frobenius norm
- The vector p-norm for $p < \infty$
- The quantum Renyi divergence $\tilde{D}_\alpha$ when $\alpha < \infty$ (defined in [MLDS+13]), including the relative entropy

I have also found proofs that the following quantities **do** yield progress functions (see Theorem A.2 and Theorem A.3 for details):

- The vector ∞-norm (also called the max norm)
- The γ_2 norm

Unfortunately, neither yields an interesting new adversary method. The max norm adversary always returns 1, while the γ_2 norm adversary is strictly weaker than the negative weight adversary.

I was not able to find conclusive results either way for several other matrix norms, including the entrywise $(2, \infty)$-norm and most operator norms.

8.2 Geodesic distances

One interesting potential progress function comes from the study of Riemannian geometry. If you give $\mathrm{Pos}(\mathcal{H})$ a Riemannian metric tensor, then you can define the distance between pairs of positive matrices as the length of the geodesic connecting those matrices. This quantity is called the geodesic distance or the Fisher metric [LSY19], and it can be computed by

$$\delta_2(X, Y) = \left\|\log(\lambda(XY^{-1}))\right\|_2$$

where λ returns the vector of eigenvalues, and log is a vector function that acts on each entry independently. This definition is quite similar to max-relative entropy, in particular

its symmetrized form, which equals

$$\delta_\infty(X, Y) = \max\{\mathrm{D}_{\max}(X\|Y), \mathrm{D}_{\max}(Y\|X)\} = \left\|\log(\lambda(XY^{-1}))\right\|_\infty$$

I define a generalization of the above two distances, which I call the order-p geodesic distance. For $p \in [1, \infty]$, the order-p geodesic distance from X to Y is given by

$$\delta_p(X, Y) = \left\|\log(\lambda(XY^{-1}))\right\|_p$$

The order-p geodesic distance satisfies a number of elegant properties. It is affine-invariant, meaning that $\delta_p(X, Y) = \delta_p(AXA^*, AYA^*)$ for any A. It can be reduced to a metric on the set of positive matrices that measures their distance to the identity, by defining $\delta_p(X, Y) = \delta_p(XY^{-1}, \mathbb{I}) := \delta_p(XY^{-1})$. Finally, δ_p approaches infinity as X or Y approaches the boundary of $\mathrm{Pos}(\mathcal{H})$, so it's a good representation of the geometry of the space.

I hypothesize that δ_p is a progress function. I have a proof that δ_p satisfies the triangle property (see Theorem A.5), but I have not been able to prove that it satisfies the mixing property. I have performed extensive numerical experiments and I have not yet found a counterexample to show that δ_p does not satisfy the mixing property for any p, so this line of research seems very promising for future work.

If δ_2 is indeed a progress function, then the corresponding adversary method may be useful in practice for proving lower bounds. For one, the new adversary method would be strongly theoretically motivated through the connection with Riemannian geometry. In addition, we know that δ_∞ gives a variant of the multiplicative adversary, so we already know that a closely related adversary method is very useful in practice. I conjecture that δ_2 will yield a useful new adversary method.

References

[Amb02] Andris Ambainis. Quantum lower bounds by quantum arguments. *Journal of Computer and System Sciences*, 64(4):750–767, 2002.

[Amb06] Andris Ambainis. Polynomial degree vs. quantum query complexity. *Journal of Computer and System Sciences*, 72(2):220–238, 2006.

[Amb18] Andris Ambainis. Understanding quantum algorithms via query complexity. In *Proceedings of the International Congress of Mathematicians: Rio de Janeiro 2018*, pages 3265–3285. World Scientific, 2018.

[AMRR11] Andris Ambainis, Loïck Magnin, Martin Roetteler, and Jeremie Roland. Symmetry-assisted adversaries for quantum state generation. *2011 IEEE 26th Annual Conference on Computational Complexity*, Jun 2011.

[And94] Tsuyoshi Ando. Majorizations and inequalities in matrix theory. *Linear algebra and its Applications*, 199:17–67, 1994.

[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, 1998.

[BDF+11] Dan Boneh, Özgür Dagdelen, Marc Fischlin, Anja Lehmann, Christian Schaffner, and Mark Zhandry. Random oracles in a quantum world. In *International conference on the theory and application of cryptology and information security*, pages 41–69. Springer, 2011.

[BPSW06] Paul Beame, Toniann Pitassi, Nathan Segerlind, and Avi Wigderson. A strong direct product theorem for corruption and the multiparty communication complexity of disjointness. *computational complexity*, 15(4):391–432, 2006.

[BRWY13] Mark Braverman, Anup Rao, Omri Weinstein, and Amir Yehudayoff. Direct products in communication complexity. In *2013 IEEE 54th Annual Symposium on Foundations of Computer Science*, pages 746–755. IEEE, 2013.

[BSS01] Howard Barnum, Michael Saks, and Mario Szegedy. Quantum decision trees and semidefinite programming. Technical report, Los Alamos National Lab.(LANL), Los Alamos, NM (United States), 2001.

[Dat09] Nilanjana Datta. Min- and max-relative entropies and a new entanglement monotone. *IEEE Transactions on Information Theory*, 55(6):2816–2826, Jun 2009.

[Dru12] Andrew Drucker. Improved direct product theorems for randomized query complexity. *computational complexity*, 21(2):197–244, 2012.

[HLS07] Peter Hoyer, Troy Lee, and Robert Spalek. Negative weights make adversaries stronger. In *Proceedings of the thirty-ninth annual ACM symposium on Theory of computing*, pages 526–535, 2007.

[JK22] Rahul Jain and Srijita Kundu. A direct product theorem for quantum communication complexity with applications to device-independent qkd. In *2021 IEEE 62nd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 1285–1295. IEEE, 2022.

[LMRŠ10] Troy Lee, Rajat Mittal, Ben W Reichardt, and Robert Špalek. An adversary for algorithms. *arXiv preprint arXiv:1011.3020*, pages 1–18, 2010.

[LMSS07] Nati Linial, Shahar Mendelson, Gideon Schechtman, and Adi Shraibman. Complexity measures of sign matrices. *Combinatorica*, 27(4):439–463, 2007.

[LR12] Troy Lee and Jeremie Roland. A strong direct product theorem for quantum query complexity. *2012 IEEE 27th Conference on Computational Complexity*, Jun 2012.

[LSŠ08] Troy Lee, Adi Shraibman, and Robert Špalek. A direct product theorem for discrepancy. In *2008 23rd Annual IEEE Conference on Computational Complexity*, pages 71–80. IEEE, 2008.

[LSY19] Lek-Heng Lim, Rodolphe Sepulchre, and Ke Ye. Geometric distance between positive definite matrices of different dimensions. *IEEE Transactions on Information Theory*, 65(9):5401–5405, 2019.

[MLDS+13] Martin Müller-Lennert, Frédéric Dupuis, Oleg Szehr, Serge Fehr, and Marco Tomamichel. On quantum rényi entropies: A new generalization and some properties. *Journal of Mathematical Physics*, 54(12):122203, 2013.

[MR15] Loïck Magnin and Jérémie Roland. Explicit relation between all lower bound techniques for quantum query complexity. *International Journal of Quantum Information*, 13(04):1350059, 2015.

[NC10] Michael A. Nielsen and Isaac L. Chuang. *Quantum Computation and Quantum Information: 10th Anniversary Edition*. Cambridge University Press, 2010.

[RT19] Ran Raz and Avishay Tal. Oracle separation of bqp and ph. In *Proceedings of the 51st annual ACM SIGACT symposium on theory of computing*, pages 13–23, 2019.

[San08] Miklos Santha. Quantum walk based search algorithms. In *International Conference on Theory and Applications of Models of Computation*, pages 31–46. Springer, 2008.

[She12] Alexander A Sherstov. Strong direct product theorems for quantum communication and query complexity. *SIAM Journal on Computing*, 41(5):1122–1165, 2012.

[Špa08] Robert Špalek. The multiplicative quantum adversary. In *2008 23rd Annual IEEE Conference on Computational Complexity*, pages 237–248. IEEE, 2008.

[ŠS05] Robert Špalek and Mario Szegedy. All quantum adversary methods are equivalent. In *International Colloquium on Automata, Languages, and Programming*, pages 1299–1311. Springer, 2005.

[Zha05] Shengyu Zhang. On the power of ambainis lower bounds. *Theoretical Computer Science*, 339(2-3):241–256, 2005.

Appendix A

Additional Proofs

This appendix gives the proofs of various technical lemmas and claims that were omitted from the main body of this thesis.

Lemma A.1. *For any Gram matrices σ_1, σ_2, we have*

$$\begin{aligned}\mathrm{Aadv}(\sigma_1, \sigma_2) &= \max_{\Gamma \in \mathrm{Pos}(\mathcal{H})} \frac{\|\Gamma \circ (\sigma_1 - \sigma_2)\|}{\max_i \|\Gamma \circ (J - S_i)\|} \\ &= \max_{\Gamma \in \mathrm{Herm}(\mathcal{H})} \frac{\|\Gamma \circ (\sigma_1 - \sigma_2)\|}{\max_i \|\Gamma \circ (J - S_i)\|}\end{aligned}$$

where $\mathrm{Herm}(\mathcal{H})$ *is the set of all Hermitian matrices on* $\mathcal{H}$*. Furthermore, if* σ_1 *and* σ_2 *are both real symmetric, then we also have*

$$\mathrm{Aadv}(\sigma_1, \sigma_2) = \max_{\Gamma \in \mathrm{Sym}(\mathcal{H})} \frac{\|\Gamma \circ (\sigma_1 - \sigma_2)\|}{\max_i \|\Gamma \circ (J - S_i)\|}$$

where $\mathrm{Sym}(\mathcal{H})$ *is the set of all real symmetric matrices on* $\mathcal{H}$*. Note that the supremum in the original definition of* Aadv *has been replaced by a maximum, which means that the maximum* Γ *is attained in every case.*

Proof. I first show that the maximum is attained in the case where $\Gamma \in \mathrm{Pos}(\mathcal{H})$. The general proof strategy is to reduce the optimization problem to optimizing a continuous function over a compact domain. Then by the extreme value theorem, we can conclude the maximum is indeed attained.

Let Γ be a candidate solution to $\text{Aadv}(\sigma_1, \sigma_2)$. We can assume without loss of generality that $\max_i \|\Gamma \circ (J - S_i)\| = 1$, as we can scale Γ by any constant without affecting the objective value. This eliminates the case where the denominator goes to 0, which guarantees we are optimizing a continuous function. We can also assume that the diagonal elements of Γ are all 0, as the diagonal elements of $\sigma_1 - \sigma_2$ and $J - S_i$ are all 0, so the diagonal elements of Γ are destroyed by the Hadamard product regardless. Finally, note that for any non-diagonal entry, there is some i for which $J - S_i$ is equal to 2 on that entry (as any distinct bit strings x, y must disagree on some bit i). Since the spectral norm is at least the max norm, we have $\max_i \|\Gamma \circ (J - S_i)\| \geq 2 \cdot \|\Gamma\|_{max}$, so $\|\Gamma\|_{max} \leq \frac{1}{2}$. Note that the set of matrices Γ satisfying $\|\Gamma\|_{max} \leq \frac{1}{2}$ is a compact ball. Together with the other conditions on Γ, we are optimizing Γ over an intersection of several closed and compact sets, so Γ is optimized over a compact set, so the maximum is attained. Thus, we have

$$\text{Aadv}(\sigma_1, \sigma_2) = \max_{\Gamma \in \text{Pos}(\mathcal{H})} \frac{\|\Gamma \circ (\sigma_1 - \sigma_2)\|}{\max_i \|\Gamma \circ (J - S_i)\|}$$

Next, I show that we can relax the condition $\Gamma \in \text{Pos}(\mathcal{H})$ to $\Gamma \in \text{Herm}(\mathcal{H})$ without affecting the objective value. We know that $\text{Pos}(\mathcal{H}) \subset \text{Herm}(\mathcal{H})$, so we clearly have

$$\sup_{\Gamma \in \text{Pos}(\mathcal{H})} \frac{\|\Gamma \circ (\sigma_1 - \sigma_2)\|}{\max_i \|\Gamma \circ (J - S_i)\|} \leq \sup_{\Gamma \in \text{Herm}(\mathcal{H})} \frac{\|\Gamma \circ (\sigma_1 - \sigma_2)\|}{\max_i \|\Gamma \circ (J - S_i)\|}$$

We just need to prove the other direction of the inequality. Let $\Gamma \in \text{Herm}(\mathcal{H})$ be a candidate matrix for the right hand side. Then let $\Gamma' = \Gamma - \lambda_{min}(\Gamma) \cdot \mathbb{I}$. Note that $\lambda_{min}(\Gamma') = \lambda_{min}(\Gamma) - \lambda_{min}(\Gamma) \cdot 1 = 0$, so all the eigenvalues of Γ' are non-negative, so $\Gamma' \in \text{Pos}(\mathcal{H})$. Then, note that Γ and Γ' only differ on their diagonal entries, which are discarded by the Hadamard products with $\sigma_1 - \sigma_2$ and $J - S_i$. Thus we have $\Gamma \circ (\sigma_1 - \sigma_2) = \Gamma' \circ (\sigma_1 - \sigma_2)$ and $\Gamma \circ (J - S_i) = \Gamma' \circ (J - S_i)$, so Γ' attains the same objective value as Γ. This proves that optimizing over $\text{Pos}(\mathcal{H})$ is at least as strong as optimizing over $\text{Herm}(\mathcal{H})$, so we have

$$\sup_{\Gamma \in \text{Pos}(\mathcal{H})} \frac{\|\Gamma \circ (\sigma_1 - \sigma_2)\|}{\max_i \|\Gamma \circ (J - S_i)\|} \geq \sup_{\Gamma \in \text{Herm}(\mathcal{H})} \frac{\|\Gamma \circ (\sigma_1 - \sigma_2)\|}{\max_i \|\Gamma \circ (J - S_i)\|}$$

Combining this with the above inequality proves the two sides are equal. Since we have already proved the maximum is attained for the left side, the maximum must also be attained for the right side, so we have

$$\max_{\Gamma \in \text{Pos}(\mathcal{H})} \frac{\|\Gamma \circ (\sigma_1 - \sigma_2)\|}{\max_i \|\Gamma \circ (J - S_i)\|} = \max_{\Gamma \in \text{Herm}(\mathcal{H})} \frac{\|\Gamma \circ (\sigma_1 - \sigma_2)\|}{\max_i \|\Gamma \circ (J - S_i)\|}$$

I now prove the final claim in the lemma. The general strategy is to write Aadv as a semidefinite program with entirely real constraints and only one complex matrix variable. Then we replace this complex matrix with a real symmetric matrix and construct an equivalent solution to the original problem.

Assume σ_1 and σ_2 are both real symmetric. We can assume without loss of generality that $\max_i \|\Gamma \circ (J - S_i)\| \leq 1$, as we can scale Γ by any constant without affecting the objective value. We can replace the denominator of our objective value with this condition to rewrite our optimization problem as the following:

$$\begin{aligned} \text{Aadv}(\sigma_1, \sigma_2) = \quad & \max_{\Gamma \in \text{Herm}(\mathcal{H})} && \|\Gamma \circ (\sigma_1 - \sigma_2)\| \\ & \text{subject to} && \|\Gamma \circ (J - S_i)\| \leq 1 \text{ for all } i \in [n] \end{aligned}$$

We can assume without loss of generality that the primary eigenvalue of $\Gamma \circ (\sigma_1 - \sigma_2)$ is positive, as we can simply negate Γ otherwise. Then we have $\|\Gamma \circ (\sigma_1 - \sigma_2)\| = \max_\delta \langle \delta\delta^*, \Gamma \circ (\sigma_1 - \sigma_2)\rangle$, where δ is a unit vector. We also know that $\|\Gamma \circ (J - S_i)\| \leq 1$ if and only if $-\mathbb{I} \leq \Gamma \circ (J - S_i) \leq \mathbb{I}$. Using these, we can replace the above problem with

$$\begin{aligned} \text{Aadv}(\sigma_1, \sigma_2) = \quad & \max_{\Gamma \in \text{Herm}(\mathcal{H})} \max_{\delta} && \langle \delta\delta^*, \Gamma \circ (\sigma_1 - \sigma_2)\rangle \\ & \text{subject to} && -\mathbb{I} \leq \Gamma \circ (J - S_i) \leq \mathbb{I} \text{ for all } i \in [n] \end{aligned}$$

If we replace the maximum over δ with a supremum, we can also assume that δ has no zero entries, as the supremum lets us get arbitrary close to zero entries. Then note that $\delta\delta^*$ is positive, so by the Schur product theorem $0 \leq X - Y$ iff $0 \leq (X - Y) \circ \delta\delta^*$, so the above is equivalent to

$$\begin{aligned} \text{Aadv}(\sigma_1, \sigma_2) = \quad & \max_{\Gamma \in \text{Herm}(\mathcal{H})} \sup_{\delta} && \langle \delta\delta^* \circ \Gamma, \sigma_1 - \sigma_2\rangle \\ & \text{subject to} && -\delta\delta^* \circ \mathbb{I} \leq \delta\delta^* \circ \Gamma \circ (J - S_i) \leq \delta\delta^* \circ \mathbb{I} \text{ for all } i \in [n] \end{aligned}$$

Then let $Z = \delta\delta^* \circ \Gamma$ and $\Delta = \delta\delta^* \circ \mathbb{I}$. We can change variables by optimizing over all $Z \in \text{Herm}(\mathcal{H})$ instead of over all $\Gamma \in \text{Herm}(\mathcal{H})$. Then Δ can be set to any real diagonal matrix with positive entries, so we get

$$\begin{aligned} \text{Aadv}(\sigma_1, \sigma_2) = \quad & \max_{Z \in \text{Herm}(\mathcal{H})} \sup_{\Delta} && \langle Z, \sigma_1 - \sigma_2\rangle \\ & \text{subject to} && -\Delta \leq Z \circ (J - S_i) \leq \Delta \text{ for all } i \in [n] \end{aligned}$$

We have finally reduced the problem to a semi-definite program with entirely real constraints and only a single complex variable. Now, since all the matrices except Z are real symmetric, we can replace Z with Z^T to get another valid solution with the same objective value. In fact, any convex combination of Z and Z^T yields a valid solution with the same objective value, as both the objective value and the constraints are entirely linear. Thus $Z' = \frac{Z+Z^T}{2}$ is real symmetric and achieves at least the same objective value as Z, so we can constrain the problem to only optimize over real symmetric matrices without decreasing the objective function. This change also can't increase the objective function (as $\mathrm{Sym}(\mathcal{H}) \subset \mathrm{Herm}(\mathcal{H})$), so we have

$$\begin{aligned} \mathrm{Aadv}(\sigma_1, \sigma_2) = \quad & \max_{Z \in \mathrm{Sym}(\mathcal{H})} \sup_{\Delta} \quad \langle Z, \sigma_1 - \sigma_2 \rangle \\ & \text{subject to} \qquad -\Delta \leq Z \circ (J - S_i) \leq \Delta \text{ for all } i \in [n] \end{aligned}$$

Then, we perform the chain of equivalences in reverse order to return to the original problem. We pick some real unit vector δ such that $\delta\delta^* = \Delta$, then change variables back to Γ, δ, where Γ is the unique matrix with $\Gamma \circ \delta\delta^* = Z$ (which must exist as δ has entirely nonzero entries). We then convert the inner product and positivity conditions back into spectral norms, and finally relax the bound on the denominator to get

$$\mathrm{Aadv}(\sigma_1, \sigma_2) = \max_{\Gamma \in \mathrm{Sym}(\mathcal{H})} \frac{\|\Gamma \circ (\sigma_1 - \sigma_2)\|}{\max_i \|\Gamma \circ (J - S_i)\|}$$

This finally completes the proof of the lemma. □

Theorem A.2. *The function $d(X \to Y) = \|X - Y\|_{max}$ is a progress function, but the adversary method Adv_{max} it generates is not very useful, as $Adv_{max}(\sigma_1, \sigma_2) \leq 1$.*

Proof. I show that d satisfies each of the properties of a progress function.

1. Triangle: This follows immediately from the triangle property of the max norm.
2. Mixing: We have

$$
\begin{aligned}
d(\sum_i X_i \circ Z_i \to \sum_i Y_i \circ Z_i) &= \left\| \sum_i (X_i - Y_i) \circ Z_i \right\|_{max} \\
&= \max_{x,y} \left| \sum_i (X_i - Y_i)[x,y] \cdot Z_i[x,y] \right| \\
&\leq \max_{x,y} \sum_i |(X_i - Y_i)[x,y] \cdot Z_i[x,y]| \\
&\leq \max_{x,y} \max_i |(X_i - Y_i)[x,y]| \cdot \sum_i |Z_i[x,y]| \\
&\leq \max_{x,y} \max_i |(X_i - Y_i)[x,y]| \cdot \max_{x,y} \sum_i |Z_i[x,y]| \\
&= \max_i \|X_i - Y_i\|_{max} \cdot \max_{x,y} \sum_i |Z_i[x,y]| \\
&\leq \max_i \|X_i - Y_i\|_{max} \\
&= \max_i d(X_i \to Y_i)
\end{aligned}
$$

where we know that $\max_{x,y} \sum_i |Z_i[x,y]| \leq 1$ because we can assume $Z_i[x,y]$ is positive (as we can Hadamard it by the appropriate S_j otherwise), and since $\sum_i Z_i$ is a Gram matrix, we have $\|\sum_i Z_i\|_{max} \leq 1$.

Since the max norm is a progress function, it gives us a new adversary method

$$
Adv_{max}(\sigma_1, \sigma_2) = \sup_{\Gamma \in \mathrm{Pos}(\mathcal{H})} \frac{\|\Gamma \circ (\sigma_1 - \sigma_2)\|_{max}}{\max_i \|\Gamma \circ (J - S_i)\|_{max}}
$$

Note that $\max_i \|\Gamma \circ (J - S_i)\|_{max} = 2 \|\Gamma \circ (J - \mathbb{I})\|_{max}$, as for every nondiagonal entry of Γ, there is always some i for which $J - S_i$ has a 2 in that entry. We also know that

$$
\|\Gamma \circ (\sigma_1 - \sigma_2)\|_{max} \leq \|\Gamma \circ (J - \mathbb{I})\|_{max} \cdot \|\sigma_1 - \sigma_2\|_{max} \leq 2 \|\Gamma \circ (J - \mathbb{I})\|_{max}
$$

Therefore, we have

$$
Adv_{max}(\sigma_1, \sigma_2) \leq \sup_{\Gamma \in \mathrm{Pos}(\mathcal{H})} \frac{2 \|\Gamma \circ (J - \mathbb{I})\|_{max}}{2 \|\Gamma \circ (J - \mathbb{I})\|_{max}} = 1
$$

Since we trivially have $Q^c(\sigma_1, \sigma_2) \geq 1$ whenever $\sigma_1 \neq \sigma_2$, this bound is useless. □

Theorem A.3. *The function $d(X \to Y) = \gamma_2(X - Y)$ is a progress function, but the adversary method Adv_{γ_2} it generates is not very useful, as $Adv_{\gamma_2}(\sigma_1, \sigma_2) \leq \mathrm{Aadv}(\sigma_1, \sigma_2)$.*

Proof. I show that d satisfies each of the properties of a progress function.

1. Triangle: This follows immediately from the triangle property of the γ_2 norm.
2. Mixing: We have

$$d(\sum_i X_i \circ Z_i \to \sum_i Y_i \circ Z_i) = \gamma_2 \left(\sum_i (X_i - Y_i) \circ Z_i \right) = \max_{Q:\|Q\|\leq 1} \left\| \sum_i Q \circ (X_i - Y_i) \circ Z_i \right\|$$

By the mixing property of the spectral norm, this spectral norm is smaller than

$$\max_{Q:\|Q\|\leq 1} \left\| \sum_i Q \circ (X_i - Y_i) \circ Z_i \right\| \leq \max_{Q:\|Q\|\leq 1} \max_i \|Q \circ (X_i - Y_i)\| = \max_i \gamma_2(X_i - Y_i)$$

Thus the γ_2 norm is also a progress function. This proof can be viewed as an alternate version of Theorem 5.4, as the argument above generalizes to show that if d is a progress function, then $d'(X \to Y) = \max_{Q:\|Q\|\leq 1} d(Q \circ X \to Q \circ Y)$ is also a progress function.

Since the γ_2 norm is a progress function, it gives us a new adversary method

$$Adv_{\gamma_2}(\sigma_1, \sigma_2) = \sup_{\Gamma \in \mathrm{Pos}(\mathcal{H})} \frac{\gamma_2(\Gamma \circ (\sigma_1 - \sigma_2))}{\max_i \gamma_2(\Gamma \circ (J - S_i))}$$

However, this adversary method is weaker than the additive adversary. Using the definition of the γ_2 norm in terms of the spectral norm, we have

$$\begin{aligned} Adv_{\gamma_2}(\sigma_1, \sigma_2) &= \sup_{\Gamma \in \mathrm{Pos}(\mathcal{H})} \frac{\max_{Q:\|Q\|=1} \|Q \circ \Gamma \circ (\sigma_1 - \sigma_2)\|}{\max_i \max_{Q:\|Q\|=1} \|Q \circ \Gamma \circ (J - S_i)\|} \\ &\leq \sup_{\Gamma \in \mathrm{Pos}(\mathcal{H})} \max_{Q:\|Q\|=1} \frac{\|Q \circ \Gamma \circ (\sigma_1 - \sigma_2)\|}{\max_i \|Q \circ \Gamma \circ (J - S_i)\|} \\ &\leq \sup_{\Gamma \in \mathrm{Herm}(\mathcal{H})} \frac{\|\Gamma \circ (\sigma_1 - \sigma_2)\|}{\max_i \|\Gamma \circ (J - S_i)\|} \\ &= \mathrm{Aadv}(\sigma_1, \sigma_2) \end{aligned}$$

Since this method is both weaker and more complicated than the additive adversary method, I expect that it will not have any uses. □

To prove the following theorem, I need a small lemma related to majorization of vectors. As a reminder, majorization is defined as follows: let v, u be vectors in $\mathbb{R}^n$ whose entries are sorted in descending order. We say that v majorizes u (written $v \succ u$) if $\sum_{i=1}^k v_i \geq \sum_{i=1}^k u_i$ for all $k \in 1 \ldots n$ and $\sum_{i=1}^n v_i = \sum_{i=1}^n u_i$.

Lemma A.4. *If X, Y are positive, then $\log(\lambda(X) \cdot \lambda(Y)) \succ \log(\lambda(XY))$, where λ returns the vector of eigenvalues, and $\log$ and $\cdot$ are vector functions that act on each entry independently.*

Proof. See equation (5.12) of [And94]. □

Theorem A.5. *For any $p \in [1, \infty]$ and any positive matrices X, Y, Z, the order-p geodesic distance satisfies $\delta_p(X, Z) \leq \delta_p(X, Y) + \delta_p(Y, Z)$.*

Proof. Recall that $\delta_p(X, Y) = \left\|\log(\lambda(XY^{-1}))\right\|_p$. Also recall that for any A, B, we have $\lambda(AB) = \lambda(BA)$. From this equation, we can show several basic properties of δ_p:

1. $\delta_p(X, Y) = \left\|\log(\lambda(XY^{-1}))\right\|_p = \left\|\log(\lambda(Y^{-1}X))\right\|_p = \delta_p(Y^{-1}, X^{-1})$
2. $\delta_p(X, Y) = \left\|\log(\lambda(XY^{-1}))\right\|_p = \left\|\log(\lambda(Y^{-1/2}XY^{-1/2}))\right\|_p = \delta_p(Y^{-1/2}XY^{-1/2}, \mathbb{I})$
3. $\delta_p(AXA^*, AYA^*) = \left\|\log(\lambda(AXA^{*-1}Y^{-1}A^{-1}))\right\|_p = \left\|\log(\lambda(XY^{-1}))\right\|_p = \delta_p(X, Y)$

Let X, Y be arbitrary positive matrices. Since the vector p-norm is Schur convex, Lemma A.4 implies that $\left\|\log(\lambda(X) \cdot \lambda(Y))\right\|_p \geq \left\|\log(\lambda(XY))\right\|_p$. By the triangle inequality, we have $\left\|\log(\lambda(X))\right\|_p + \left\|\log(\lambda(Y))\right\|_p \geq \left\|\log(\lambda(XY))\right\|_p$. Applying the definition of δ_p, we conclude that $\delta_p(X, \mathbb{I}) + \delta_p(Y, \mathbb{I}) \geq \delta_p(X, Y^{-1})$.

Then, we use all the properties above to prove that

$$\begin{aligned}
\delta_p(X, Y) + \delta_p(Y, Z) &= \delta_p(X, Y) + \delta_p(Z^{-1}, Y^{-1}) \\
&= \delta_p(Y^{-1/2}XY^{-1/2}, \mathbb{I}) + \delta_p(Y^{1/2}Z^{-1}Y^{1/2}, \mathbb{I}) \\
&\geq \delta_p(Y^{-1/2}XY^{-1/2}, (Y^{1/2}Z^{-1}Y^{1/2})^{-1}) \\
&= \delta_p(Y^{-1/2}XY^{-1/2}, Y^{-1/2}ZY^{-1/2}) \\
&= \delta_p(X, Z)
\end{aligned}$$

Therefore δ_p satisfies the triangle property. □