Optimal Success Bounds for Single Query Quantum Algorithms Computing the General SUM Problem

by

Alexander Valtchev

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

Waterloo, Ontario, Canada, 2015

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

In this thesis the problem of computing the sum of a string of arbitrary finite length drawn from an arbitrary finite alphabet is treated. The resource considered is the number of queries to some oracle which hides the string and gives access to each of its digits. The sum of a string is defined as adding all of the string's digits together modulo the alphabet size.

Classically, this problem is straightforward as any less than a number of queries equal to the string length reveals no useful information about the sum of the string. In the quantum information setting however, things are not so clear. When the alphabet size is equal to two, the problem becomes finding the parity of a bit string. This is a seminal result in quantum computation that allows a correct answer with certainty by making only half the queries that are classically needed. As the alphabet size increases beyond two however, less is known. There is an algorithm by Meyer and Pommersheim which computes the sum in this general setting with probability of success: $\min\left\{\frac{\lfloor \frac{n}{n-q} \rfloor}{k}, 1\right\}$ where n is the string length, k is the alphabet size, and q is the number of queries made. This algorithm has probability of success slightly above guessing when the number of queries are half the string length, and perfect probability of success when $n-1$ queries are made. The question dealt with in this thesis is whether this algorithm is optimal for the general sum case.

The problem is expressed as a semidefinite program, given for all instances. The instance for strings of length two and algorithms making a single query is solved and a proof is given. Significant insight into the multi-query case is also provided.

Acknowledgements

Thank you to my supervisors Richard Cleve, and John Watrous - with whom I have been lucky to work and learn from.

Thank you to my parents.

Thank you to my friends.

Dedication

This is dedicated to the spirit of curiosity and the courage to pursue.

Table of Contents

List of Figures

Chapter 1

Introduction

Theoretical computer science seeks to shed light on how capable certain computational models are at solving problems while minimizing the use of some computational resource. In the last two decades a lot of research has been done towards outlining the capability of a computational device operating within the mathematical framework of quantum mechanics. In numerous cases, a separation between quantum and classical computation has been shown. Most notable among these results is perhaps the problem of factoring large numbers since it directly impacts advanced techniques that aim to break RSA encryption. An algorithm operating in the quantum setting was shown that computes the factors exponentially faster than the best known classical algorithm. [15]

A particular class of problems for which separations have been shown is known as oracle or black box problems. In this setting, a black box is given which hides some information from a predefined set. An algorithm then aims to perform some computation that is dependent on the hidden item and produce a correct result within some tolerable probability of success. Access to this black box might be computationally cumbersome. As a computational resource, black box access is counted in terms of the number queries made where a single query often reveals partial information in accordance with some predefined query model. Of interest then, is how many queries an algorithm makes in order to achieve a certain probability of success for a specific problem.

The particular problem of interest in this thesis is as follows. Given a string x, of elements $(a_0, \ldots, a_{n-1})$ from the ring $\mathbb{Z}_k$ with the standard operations of addition and multiplication, compute the sum of the digits mod k. Note that the string length, n, and the alphabet size, k are known, while the particular string is unknown (and perhaps chosen in an adversarial fashion). More formally,

$$\text{Given: } x = (a_0, \ldots, a_{n-1}) \in \mathbb{Z}_k^n$$
$$\text{Compute: } \sum_{i=0}^{n-1} a_i \in \mathbb{Z}_k$$

where k and n can be any fixed, positive integers. Access to x is provided via an oracle that allows one to query each index of x, producing the value at that index.

In this thesis, we will prove optimal success bounds for the generalized sum problem for quantum algorithms which make a single query. Such algorithms consist of preparation of some initial quantum state, the query, post-processing, and measurement to produce a classical result. These algorithms mimic the strategy used to solve the case when $k = 2$ by Deutche's algorithm since only one query is made per each pair of digits in a non adaptive manner.

The remainder of the introduction chapter defines mathematical notations used throughout the thesis, a primer on semidefinite programming in the quantum computation context, and a more rigorous definition of the oracle setting in which the problem is cast.

Addendum: On March 18th, 2015, when the thesis was already complete and in final editing phases, a paper was posted to the archive that also proves the main result in this thesis through a completely different approach. Their results can be found in [4]. This thesis goes beyond the result from this paper since it uses semidefinite programming and lays a strong foundation to providing bounds for two or more query algorithms for symmetric functions.

1.1 Mathematical Notation

Before proceeding, it is time to define the notation used in this thesis. Most of the notation is taken from [17] which introduces it and contains further details on all of the concepts used in this text.

Complex Euclidean vector spaces are denoted in script letters towards the end of the alphabet: $\mathcal{X}, \mathcal{Y}, \mathcal{W}, \mathcal{Z}$. Typically a complex Euclidean vector space is of the form $\mathcal{X} = \mathbb{C}^n$ where $\dim(\mathcal{X}) = n$.

The set of all linear operators on some complex Euclidean vector space are denoted $\mathrm{L}(\mathcal{X})$.

The set of all Hermitian operators on some complex Euclidean vector space are $\mathrm{Herm}(\mathcal{X})$.

The set of all positive semidefinite operators is $\mathrm{Pos}(\mathcal{X})$.

The set of all unitary operators is $\mathrm{U}(\mathcal{X})$.

The set of all projection operators is $\mathrm{Proj}(\mathcal{X})$.

The set of all density operators is $\mathrm{D}(\mathcal{X})$.

The set of all unit vectors on some complex Euclidean vector space is $\mathrm{S}(\mathcal{X})$.

$\{e_1, \ldots, e_d\}$ - are the standard basis vectors on some d-dimensional complex Euclidean vector space.

$\oplus$ - is the direct sum between two matrices, where $A \oplus B = \begin{pmatrix} A & 0 \\ 0 & B \end{pmatrix}$. Or the sum of two group elements under the group addition. It should be clear from the context when each definition is used.

$0_d, \mathcal{J}_d$ - denote the all zeros and all ones matrix on d dimensions respectively. Sometimes the subscript is dropped when it is clear what the dimensions of a block must be.

Quantum pure states are expressed in Dirac notation as: $|\psi\rangle, |\phi\rangle$. Alternatively they are also represented as $u, v, w \in \mathcal{X}$ since they are vectors in $\mathcal{X}$. Arbitrary quantum states are represented by their density operators $\rho, \sigma, \in \mathrm{D}(\mathcal{X})$.

Linear operators that are positive semidefinite are usually denoted $A, B, C, P, Q, R, S \in \mathrm{Pos}(\mathcal{X})$. Operators in general are capital letters. Typically, variables that are linear operators are denoted as W, X, Y, Z. J is typically used for matrices that are Choi representations.

Channels are completely positive, trace preserving maps mapping between two complex Euclidean vector spaces $\mathrm{L}(\mathcal{X}) \to \mathrm{L}(\mathcal{Y})$. The set of all such channels is $\mathrm{C}(\mathcal{X}, \mathcal{Y})$, and the channels themselves are denoted as Φ,Ψ,Ξ. For $P \in \mathrm{Pos}(\mathcal{X})$ and $\Phi \in \mathrm{C}(\mathcal{X}, \mathcal{Y})$, $\Phi(P) = Q \in \mathrm{Pos}(\mathcal{Y})$ holds.

The conjugate transpose of a state or operator is u^*, ρ^*, P^*, or $\langle a|$.

The inner product of two vectors $u, v \in \mathcal{X}$ is $u^* v = \overline{v^* u} = \langle u, v\rangle = \overline{\langle v, u\rangle}$. The outer product is $uv^* = |u\rangle\langle v|$, $vu^* = |v\rangle\langle u|$. For two operators, the inner product is defined as, $\langle A, B\rangle = \mathrm{Tr}(A^* B)$.

The adjoint of a quantum channel $\Phi \in \mathrm{C}(\mathcal{X}, \mathcal{Y})$ is Φ^*, defined by $\langle Y, \Phi(X)\rangle = \langle \Phi^*(Y), X\rangle$. This is a fundamental equation that is always used to compute the dual of a map given its action one way.

$\mathbb{Z}_k$ is the cyclic group order k under addition. For two elements $a, b \in \mathbb{Z}_k$, we have that $a \oplus_k b = a + b \mod k$.

$\mathbb{Z}_k^n = \mathbb{Z}_k \times \ldots \times \mathbb{Z}_k$ is the cyclic group of order k product with itself n times.

An element $x \in \mathbb{Z}_k^n$ can be considered as a string of length n where each digit is an element of $\mathbb{Z}_k$. We think of x having the form $(a_1, \ldots, a_n)$, where a_i is the ith digit of the string - an element of $\mathbb{Z}_k$. On the other hand, if we index all k^n strings in $\mathbb{Z}_n^k$ uniquely, they are denoted as $\{x_1, \ldots, x_m\}$. Through out the text m is always taken to be k^n.

$\mathcal{A}, \mathcal{B}$ - script letters at the beginning of the alphabet are typically general sets.

1.2 Semidefinite Programming

Much of our exploration into the what the optimal probability of success for the SUM problem is for arbitrary alphabet sizes and string lengths will come from expressing this quantity as a semidefinite program (SDP) and attempting to evaluate its optimal value. Semidefinite programs are a particular class of linear optimization problems and are introduced here.

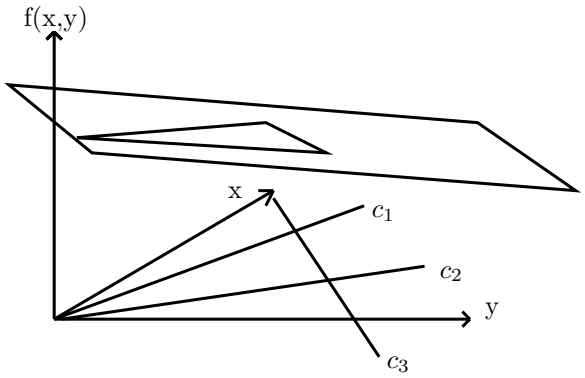

Figure 1.1: A linear constraint problem with two variables, three constraints, and objective function f.

A **linear optimization problem** (**LP**) aims to optimize the value of some linear **objective function** whose variables can vary over domains defined by a set of linear constraints. Figure 1.1 shows an example of a very simple linear program with two variables, x and y. The constraints, which are either inequality or equality conditions, restrict the

function variables to form what is called a **feasible region**. Any values assigned to x and y from the feasible region are called **feasible solutions** to the optimization problem because the objective function, f, is defined there, and all constraints are satisfied. The **optimal value** of the LP is then the minimum or maximum of f on this feasible region. Note that it is possible that the constraints are too strict and there are no feasible solutions in which case the feasible region of the LP is the empty set.

A **semidefinite program** (**SDP**) is similar, with the exception that it optimizes linear functions over positive semidefinite operators acting on some finite dimensional complex Euclidean vector space, $\mathcal{X}$. The set of all positive semidefinite operators, $\mathrm{Pos}(\mathcal{X})$, form a cone in the real vector space of all Hermitian operators, $\mathrm{Herm}(\mathcal{X})$. The linear constraints of an SDP define some subspace of $\mathrm{L}(\mathcal{X})$. The purpose of an SDP is to optimize over the intersection of this subspace with the positive semidefinite cone. Figure 1.2 shows a picture in low dimensions which may be useful for building intuition.

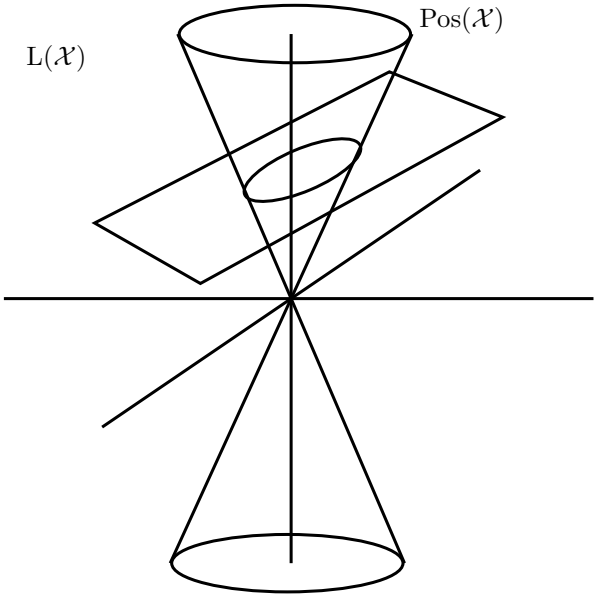

Figure 1.2: A three dimensional depiction of what is happening. The entire space would be $\mathrm{L}(\mathcal{X})$ - the space of all linear operators, and the cone would be all positive semidefinite operators. The plane represents the linear constraints of the SDP.

For SDPs, the optimal solution for which the value of f is maximized can be outside the set of feasible solutions. For example, let $\mathcal{A} \neq \emptyset$ be the set of feasible solutions based on some constraints of an LP. It is possible that f achieves its optimal value, α at $A \notin \mathcal{A}$.

So there exists some sequence $\{A_n\} \in \mathcal{A}$ that converges to a and $f(A_n) \to \alpha$ as $n \to \infty$. If the optimal solution of f lies inside the feasible region then we say that the optimum is **achievable**, and the supremum (or infimum) can be considered to be a maximum (or minimum).

SDPs come with a powerful duality theory which makes them wonderful tools for analysis. Maximizing any linear objective function over a feasible region of some space can also be expressed as a minimization of a dual function over the feasible region of a dual space. These complementary versions of the same optimization problem are called the **primal** and the **dual**. In this text, the primal always refers to the maximization problem, while the dual refers to the complementary minimization problem. Any semidefinite program enjoys a property called weak duality between its primal and dual. Weak duality means that the optimal value of the primal is always less than or equal to the optimal value of the dual.

An SDP may come in many different forms depending on the problem at hand, however we will define a generic form and examine some properties first. Keeping such a general form in mind is beneficial for two reasons. First, it allows us to state some powerful results from the theory of SDPs. Second, any problem we suspect can be expressed as an SDP can be translated to this general form so that the tools discussed here can be applied directly. For example, one can use this general form as a guide to compute the dual of a problem to which we have the primal. Knowing both dual and primal versions is often what enables us to use SDPs as a tool for analyzing bounds. The general form of an SDP is taken from [17] and is defined as follows. A semidefinite program is: A triple, (Φ, A, B), where $\Phi : \mathrm{L}(\mathcal{X}) \to \mathrm{L}(\mathcal{Y})$, $A \in \mathrm{Herm}\,(\mathcal{X})$ and $B \in \mathrm{Herm}\,(\mathcal{Y})$. The primal problem is then

$$\begin{aligned} &\textbf{primal:} && \\ &\text{maximize} && \langle A, X \rangle \\ &\text{subject to} && \Phi(X) = B \\ &&& X \in \mathrm{Pos}(\mathcal{X}). \end{aligned} \tag{1}$$

where X is the variable being optimized over. The corresponding dual is

$$\begin{aligned} &\textbf{dual:} && \\ &\text{minimize} && \langle B, Y \rangle \\ &\text{subject to} && \Phi^*(Y) \geq A \\ &&& Y \in \mathrm{Herm}(\mathcal{Y}). \end{aligned} \tag{2}$$

where Φ^* is the adjoint map of Φ and Y is the variable being optimized over.

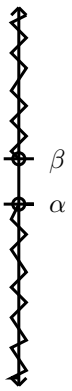

Figure 1.3: The feasible solution objective values for the primal and dual of an sdp. α and β are the respective optima, which may or may not be achievable even though both feasible regions are not empty. If strong duality holds there is no gap between α and β. If both of Slater's conditions hold, there are solutions to both the primal and dual which yield this optimal value.

An additional property some SDPs can have is strong duality. Strong duality means that the dual optimum value is equal to the primal optimum value as shown in fig 1.3. We can also check to see if there exist respective feasible solutions that achieve each optimum. Slater's Theorem for semidefinite programs gives conditions for an SDP that imply strong duality holds and optima can be achieved.

Slater's Theorem for SDPs: (3)

For an SDP specified by a triple (Φ, A, B), where $\mathcal{A}$ and $\mathcal{B}$ are the primal and dual feasible regions,

1. If $\mathcal{A} \neq \emptyset$ and $\exists\ Y\ \mathrm{Herm}(\mathcal{Y})$ where $\Phi^*(Y) > A$
   Then $\alpha = \beta$ and $\exists\ X \in \mathcal{A}$ such that $\langle A, X\rangle = \alpha$
2. If $\mathcal{B} \neq \emptyset$ and $\exists\ X > 0$ where $\Phi(X) = B$
   Then $\alpha = \beta$ and $\exists\ Y \in \mathcal{B}$ such that $\langle B, Y\rangle = \beta$

The way we have expressed Slater's theorem corresponds to the primal and dual standard form of a semidefinite program in (1) and (2) respectively. The first property states that if there is primal feasibility and strict dual feasibility (since $\Phi^*(Y)$ has to now be strictly greater than A) then there is strong duality and the primal optimum is achievable. The second property states that if there is dual feasibility and strict primal feasibility (since X

has to be strictly positive) then there is strong duality and the dual optimal is achievable. Typically, when showing that an SDP has strong duality, we want to check both of these properties and know that both optimal are achievable.

In the context of quantum information theory, SDPs are very useful since they optimize over positive semidefinite operators. Positive semidefinite operators represent arbitrary states and permissible channels acting on them, therefore SDPs are a powerful tool for analysis for many interesting questions in this setting. For instance, what start state and measurement scheme can be used to maximize the probability of knowing what channel out of some set has been given. Furthermore, restrictions can be placed via the constraints of an SDP that further narrow the problem and model desired behavior that matches the quantum setting. In some cases, solutions to the primal yield actual solutions to the quantum information problem (for example optimal sets of states, or channels that solve a problem which the SDP models). In addition, *any* solution to the primal or dual can yield an upper or lower bound on some relationship being optimized over. Of course one would have to be clever about how the objective function is defined to ensure the desired quantity is being optimized. These concepts are used in chapter three to shed light on the main problem in this thesis.

1.3 The Oracle Setting

In addition to time and space bounds on algorithms that solve certain problems within a computational framework, there is also a notion of the number of oracle or black box access a solution involves. In this setting, a black box is given that is hiding some information. There are specific rules as to how the black box may be queried. Typically, one is concerned with the number of queries issued to the black box in order to solve some sort of problem dependent on the information that is hidden within it. A straightforward problem involving queries to a black box is finding out what information is hidden in its entirety. Additionally, promises can be made about the contents of the black box. For instance, one could promise that the black box is hiding a particular element from a previously specified set. More complicated problems can involve determining some property of the information that is hidden. In such a case, we are forced to consider whether less than full knowledge of what is hidden is sufficient since it is favorable to minimize the number of queries, and perhaps trade some probability of success. The terms black box and oracle are used interchangeably throughout this work and refer to the same thing. Although black boxes are used in a variety of different settings and represent different computational power, the problem of interest in this thesis has to do with oracles hiding strings. These strings may be bit strings

or strings of higher order alphabets. The precise oracle setting which our problem is cast within is described in the next section.

Oracles Hiding Strings Consider a black box that is hiding some string $x = (a_1, \ldots, a_n) \in \mathbb{Z}_k^n$. This string is of length n and each digit a_i is an element of the cyclic group $\mathbb{Z}_k$. n and k are known before hand, so the hidden string could be any one of $|\mathbb{Z}_k^n| = k^n$ possibilities. Agreement must be made as to how such a black box reveals the information it is hiding, or alternatively, what exactly a query does. The most general query model for hidden strings of arbitrary alphabet sizes is as follows. Let the oracle take as input some index value $i \in [n]$ (the set of all positions) and produce as output a_i, the element of $\mathbb{Z}_k^n$ in that position of the hidden string. This is fine for classical computation, however, in the quantum setting we need all operations to be reversible, including that of the black box. Therefore we use the following model, outlined in figure 1.4.

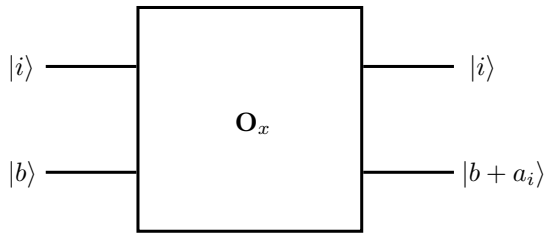

Figure 1.4: The oracle model for hiding strings of length n and alphabet k.

This black box has two inputs and two outputs. The first input is the position of the element we wish to query for. The second input is some arbitrary element of $\mathbb{Z}_k$. The black box does nothing to the first input, leaving the information about which index was queried intact. It adds the result of the query onto the second input modulo k. When using this black box classically the number of bits each input/output line needs will depend on the length of the hidden string n, and the size of the cyclic group each element is a member of, k. The same is true in the quantum setting where the size of those two constants will determine the dimension of the complex Euclidean vector space of each input/output line. This arises from the fact that each discrete element of either input line must be a basis state in some complex Euclidean vector space. For the first input and output, $|i\rangle \in \mathcal{X}_A$ is mapped to $|i\rangle \in \mathcal{Y}_A$ (where $\dim(\mathcal{X}_A) = \dim(\mathcal{Y}_A) = n$ since $i \in [n]$). For the second input and output, $|b\rangle \in \mathcal{X}_B$ is mapped to $|b \oplus a_i\rangle \in \mathcal{Y}_B$(where $\dim(\mathcal{X}_B) = \dim(\mathcal{Y}_B) = k$ since $|\mathbb{Z}_k| = k$). We can call the overall input space $\mathcal{X} = \mathcal{X}_A \otimes \mathcal{X}_B$ and the overall output space $\mathcal{Y} = \mathcal{Y}_A \otimes \mathcal{Y}_B$. Different letters for input and output are used to later keep in mind that

one is input and the other is the output space of this black box even though they have the same dimension. This dimension ($\dim(\mathcal{X}) = \dim(\mathcal{Y}) = nk$) will be central to the search for a solution of the main problem presented in this thesis. More on the precise representation of oracles hiding strings as positive semidefinite operators will be given in section chapter three, when we set up a semidefinite program for the problem.

Suppose now we are given such a black box hiding one of k^n strings for some specified n and k and are tasked with identifying precisely this string. In a classical computation setting each time a query is made, one digit of the string is uncovered. If we make $n-1$ queries, then all but one digit will be known. This narrows down the possible strings being hidden from k^n to k. In fact, each time we make a query, the number of possible strings that could be the hidden one is reduced by a factor of $\frac{1}{k}$. Given this behavior, we can consider the probability of guessing the correct answer. However, in order to correctly compute this probability we must also consider how the hidden string is *chosen* to begin with. Suppose that the oracle (representing a unique string) is chosen and given to us by some adversarial party. This party may employ any strategy they wish, from picking the same string every time, to picking uniformly at random. This additional behavior induces a prior probability on which string our oracle is hiding. If we know the adversary chooses one of two strings from the entire set which vary only one one digit, then one query will be sufficient to correctly ascertain which string it is. If they choose uniformly at random then one query, in the classical setting, will improve our chances of guessing correctly from $\frac{1}{k^n}$ to $\frac{1}{k^{n-1}}$. Of course, the only way to know the string with certainty in such a case would be to make all n queries.

Note that making queries in parallel classically may save computation time, but will not lower the number of accesses to the black box, which is the principle quantity of interest.

The problem of identifying which string an oracle is hiding is called the "oracle interrogation problem". It turns out quantum computers can do much better than classical ones in this setting. For binary strings of length n, $\lfloor \frac{n}{2} + \sqrt{n} \rfloor$ queries identify the hidden string with accuracy over 95%. The error can be made arbitrarily small by increasing the queries to $\frac{n}{2} + \mathcal{O}(\sqrt{n})$ [16].

1.4 The Problem: Summing the Digits of a Hidden String

Oracle interrogation is not the end of the story when it comes to hidden string black box problems in computing. Suppose that instead of identifying which particular string is

hidden by a black box, we wish to compute some function of the string. More precisely, for $x \in \mathbb{Z}_k^n$ hidden by an oracle we have $f : \mathbb{Z}_k^n \to \mathbb{Z}_k$, and we wish to compute $f(x)$.

If $k = 2$ then we consider boolean functions acting on binary strings of length n.

In this thesis we concern ourselves with a specific function for arbitrary k. For some hidden $x \in \mathbb{Z}_k^n$ where $x = (a_0, \ldots, a_{n-1})$, define

$$\mathrm{SUM}(x) = a_0 \oplus \cdots \oplus a_{n-1}$$

where addition is modulo k.

Let us consider some trivial examples. If $k = 1$ then $\mathrm{SUM}(x) = 0$ for any $n \geq 1$. If $k = 2$ then we have a binary string of length n and are trying to find its parity: $\mathrm{SUM}(x) \in \{0, 1\}$. This is one of the classic results in quantum information theory. In [6] a lower bound for this problem is given, and Deutsch's algorithm [11] matches it. Deutsch's algorithm uses one query to find the parity of two bits of the string. This is in turn used to find the parity of an n bit string by making one quantum query for every two digits to find the parity between them with certainty. The results are measured and the classical outcome of 0 or 1 is obtained for each pair. These outcomes are summed together classically mod 2 to find the parity of the string without making further queries. Since the one query algorithm for finding the parity of a two bit string succeeds with certainty, the final result is also certain.

Deutsch's algorithm works by querying the oracle, as defined in section 1.3, in a superposition over both digit indexes on its first input line. An oracle working as described by 1.4 over an alphabet size of two will apply either the identity or the not operator on the second qubit, depending on whether the queried index of the two bit string is a 0 or 1. This means that if a 0 state is placed on the second input line along with the superposition of both indexes on the first, the result on the second register will be a super position of both values for each digit. This is not helpful as the resultant states could be $|0\rangle, |0\rangle + |1\rangle, |1\rangle + |0\rangle, |1\rangle$ (up to a global phase) each with $\frac{1}{4}$ probability. The is no measurement strategy to distinguish perfectly between these states. However, if instead of $|0\rangle$ on the second input, $|-\rangle = |0\rangle - |1\rangle$ is placed, something different happens. $|-\rangle$ is an eigenvector of both the identity and the not operator with eigenvalue 1 and -1 respectively. Let's see what happens if the oracle is queried this way.

$$\begin{aligned}
\mathcal{O}_x(|0\rangle + |1\rangle)|-\rangle) &= \mathcal{O}_x|0\rangle|-\rangle + \mathcal{O}_x|1\rangle|-\rangle \\
&= (-1)^{a_0}|0\rangle|-\rangle + (-1)^{a_1}|1\rangle|-\rangle \\
&= ((-1)^{a_0}|0\rangle + (-1)^{a_1}|1\rangle)|-\rangle
\end{aligned}$$

The result on the second register effectively induces a partial phase on the first. Therefore, on the first register, we will have either a $|-\rangle$ if the digits are different, or $|+\rangle$ if they are the same (up to a global phase factor). Since these states are orthogonal, they are perfectly distinguishable through measurement. Thus the parity of the two bits has been found with certainty.

As we let k become arbitrarily greater than 2 but still finite, a very interesting question arises. How many queries are the minimum required to compute the sum of a string of length n with certainty? How much information do we gain per query, or alternatively, how does the probability of guessing correctly the sum increase as the number of queries range from 1 to n for some fixed n and k? Given n and k, is there a minimum number of queries for which a quantum algorithm has a probability of success greater than just taking a guess uniformly at random? Does it matter if queries are utilized within an algorithm in a non sequential non adaptive way as in Deutsch's algorithm for parity? Are there quantum algorithms that perform better for arbitrary k if they make quantum queries one after the other adapting each based on the last without measurement in between? In this thesis we aim to shed light on these questions, beginning by presenting known results on the topic.

1.5 Existing Results

At present there are two papers that contain results which provide significant insight into this problem [12, 13]. Both are by David A. Meyer and James Pommersheim. The first, pertains to the minimum queries that have to be made in order to compute functions on arbitrary finite alphabets with non-trivial probability of success. This result is referred to as the "uselessness theorem" and is outlined in detail in [12]. The uselessness theorem from that work states that for an oracle hiding a string of length n if $2q$ queries are useless classically to compute some function f, then q queries will be useless in the quantum setting. This result holds for strings of arbitrary finite alphabets, and functions that are non boolean. It provides an alternate way to arrive at the lower bound for the parity problem on binary strings and provides the same lower bound of $\lceil \frac{n}{2} \rceil$ for the general sum problem of interest here. Note that this bound coincided with the minimum useful queries which may be made. In the analysis section for making a single query, we will see why this lower bound holds due to the structure of the problem more specifically.

The second result by Meyer and Pommersheim is an explicit algorithm for the gener-

alized sum problem. For a fixed n and k this algorithm achieves probability of success

$$\min\left\{\frac{\lfloor\frac{n}{n-q}\rfloor}{k}, 1\right\}.$$

There is no proof or work showing that this algorithm is optimal for all cases of n, k, and q. It adheres to the lower bound imposed by the uselessness theorem as $\frac{n-1}{2}$ queries yield a probability of success of $\frac{1}{k}$. It also matches the probability of success of Deutsch's algorithm. Furthermore, it produces two other interesting points into view. First, $n-1$ queries always succeed in computing the general sum with probability 1 when $k \leq n$. Hence, the algorithm is optimal for this case and provides a (very small) separation between the quantum and classical scenarios. Note that if n is larger than k the algorithm is applied to every k digits to obtain the highest possible chance of success and adhere to the above bound. It also tells us that even making a single query above the uselessness theorem threshold provides an increased probability of success (of $\frac{2}{k}$) over random guessing. The existence of such an algorithm aids tremendously in outlining the general sum problem overall. It gives a lower bound on the probability of success for any fixed amount of queries q between $\lceil\frac{n}{2}\rceil$ and $n-1$. If an upper bound on this quantity which matches that of the algorithm is given by an analytic method, then optimality will be proved. The aim of exploring the general sum problem is to figure out what happens in the aforementioned query range.

The algorithm works by preparing an initial state according to some eigenvectors of the generalized not operators and performs a series of shift and query operations. The result after each query is not measured at any intermediate step, hence the queries are adaptive from one to the next, and sequential. This is a stark difference to the approach behind Deutsch's algorithm. This leaves the question of whether there is a particular threshold of what number of queries should be handled sequentially and adaptively, as apposed to measured and combined classically as in Deutsch's. The algorithm is outlined in detail in [13].

1.6 Preliminary Approaches

In this section some preliminary approaches into the sum problem are presented. Neither of them constitutes any improvement on the current results and are are included solely to give a flavor of the problem. The first, illustrates an alternate algorithm for the problem of making one query when $n = 2$. The second looks at multiple queries.

1.6.1 An Alternate Algorithm

In this section, an algorithm is presented that was devised as a preliminary insight into the problem. Let $n = 2$ and $k = 4$. Only one query can be considered here in order for the problem to stay nontrivial. Classically, one query does nothing to improve the probability to compute correct answer beyond guessing. The algorithm from [13] has a probability of success to compute the sum of an arbitrary hidden string in this setting of $\frac{1}{2}$. Let $\mathcal{U}_x$ denote some unitary transformation that functions as an oracle outlined in figure 1.4, and define the following states

$$|+\rangle = |0\rangle + |1\rangle + |2\rangle + |3\rangle$$
$$|-\rangle = |0\rangle - |1\rangle + |2\rangle - |3\rangle$$
$$|A\rangle = |0\rangle + |1\rangle - |2\rangle - |3\rangle$$
$$|B\rangle = |0\rangle - |1\rangle - |2\rangle + |3\rangle$$

where

$$|+\rangle, |-\rangle, |A\rangle, |B\rangle \in \mathbb{C}^n \text{ for } k = 4.$$

Let us see how these states behave when a generalized not operator of the form X^a for $a \in \mathbb{Z}_4$ is applied. The $|+\rangle$ remains invariant. The $|-\rangle$ state alternates between $|-\rangle$ and $-|-\rangle$ depending on whether a is even or odd. The states $|A\rangle$ and $|B\rangle$ form the following cycle per each application of X^1

$$|A\rangle \to -|B\rangle \to -|A\rangle \to |B\rangle \to |A\rangle$$

Since X^a is simply $X^1 \cdot \ldots \cdot X^1$ a times, this cycle defines the action of X for all $a \in \mathbb{Z}_k$.

If we are only looking for the parity, it amounts to finding out whether the sum of the digits is even $\{0, 2\}$ or odd $\{1, 3\}$. This can be done again in $\frac{n}{2}$ queries along the same lines as Deutsch's algorithm except by using the generalized $|-\rangle$ state on k basis vectors.

On the other hand if $(|0\rangle + |1\rangle)|A\rangle$ is used as an initial state to U_x, and Hadamard transforms are performed on the first qubit before and after the oracle acts on this input (much like in Deutsch's algorithm) the problem of finding the sum of two digits with $k = 4$ reduces to the problem of identifying which state you have been given between $|0\rangle$ and $(|0\rangle + |1\rangle)$.

1.6.2 A Mini Query Approach

One could change the query model a bit, in hopes of better understanding the problem. Suppose we are dealing with strings of length n from some finite alphabet strictly larger

than 2. Furthermore, suppose that the alphabet order is a power of 2. Then a single query to our original oracle outlined in 1.3 can be viewed as p queries to some binary oracle. This binary oracle holds the information for each digit of the hidden string in terms of bits. Suppose more precisely that $2^p = k$. Knowing one original query yields one digit, hence, p bits. Our binary oracle lets us query each one of these p bits separately, introducing the notion of fractional queries.

For a concrete example, let $k = 4$ Then each original query yields a digit from $\mathbb{Z}_4$. A binary oracle representing this string would have two binary queries per digit. So a string of length 4 would have 8 positions to query within the binary oracle model. Immediately, a weakness in this model arises. Each query on the result register must be added modulo 2 for the model to make sense. Simply knowing the parity of all pair of bits reveals the parity of the general sum, but not its actual value.

We can, however, use the oracle interrogation result mentioned in section 1.3. This allows us to find the hidden binary string with 95% accuracy at the cost of $\frac{N}{2} + \sqrt{N}$ mini queries, where $N = np$ is the total number of bits needed to represent strings with length n and alphabet k. Once we find the hidden string in binary form, all we have to do is convert each digit back to the alphabet we are dealing with, and add them all together. This means that for $k = 2^p$,

$$\frac{\frac{np}{2} + \sqrt{np}}{p}$$

queries allow us to compute the sum with 95% accuracy.

Let $p = 4$, then $k = 16$. If our strings are of length 16 too, we can identify the exact hidden string with $\frac{64}{2} + \sqrt{4 \cdot 16} = 40$ mini queries. These amount to 10 big queries for probability of success of 95%. Using the algorithm of [13], we will have a probability of success of $\frac{\lfloor \frac{16}{6} \rfloor}{16} = \frac{1}{8}$. This approach is most likely optimal for this scenario and beats the general sum algorithm, but we have drastically changed the problem. In the remainder of this work, we stick to the original oracle model, and aim at providing bounds in that setting.

The oracle interrogation algorithm could be extended to deal with strings of arbitrary alphabet sizes. Meyer and Pommersheim do this in the general sum algorithm paper and show that in that case, the strategy of finding the string with high probability and summing it classically yields a smaller probability of success for all cases in comparison to their algorithm.

Chapter 2

Optimal Bounds For Single Query Quantum Algorithms

A brief overview of the theory of semidefinite programs in the context of quantum information was given in section 1.2. It turns out that many interesting questions in quantum information theory can be posed as SDPs. They will be our main tool for determining bounds on the optimal probability of success of quantum algorithms solving the instances of the sum problem. A particular instance of the problem is defined by a given number of queries, q, a predefined string length, n, and alphabet size k. Every instance defined by unique values of these three parameters will have its own SDP associated with it. In this chapter an application of semidefinite optimization to the SUM problem for single query instances will be explored in full and an SDP will be derived for one query algorithms. Success bounds will be given through specification and analysis of a particular feasible solution for the single query case which yields a dual optimum matching the existing algorithm for the general sum of hidden strings. The SDP for multiquery algorithms will be treated in chapter three along with discussion towards bounds for the multiquery case.

2.1 SDP for the SUM Problem with 1 Query

Recall that we are given a black box hiding some string x in $\mathbb{Z}_k^n$ and we would like to compute the sum mod k of the digits of the string while making as few queries to the black box as possible. Consider what this entails in the quantum setting. The oracle is a channel whose action is given by the rules in section 1.3 applied to computational basis states of the

two input and two output registers. The parameters n and k therefore, change the input and output space dimensions of the complex Euclidean vector spaces this channel maps between. The number of queries on the other hand, each represent a call to the oracle. We begin by thinking about what happens when a single query is made by any quantum algorithm.

One oracle call means that the algorithm only gets one chance at access. The output it gets from that single query is final. There will be some initial state sent to the oracle, which will act on it and produce a resulting state. The actual initial state can be any valid quantum state of dimension greater than or equal to that of the input space of the oracle. Any additional dimensions imply ancillary qubits that we can perform operations on privately which may share entanglement with those the oracle is acting on. Since we are making only one query, once we get the result from calling the oracle on the start state, we can perform any arbitrary unitary gate on the entire system. This unitary corresponds to any post-processing that may be done by a quantum computer. After the post-processing, we must measure in some basis of choice to attain a classical result. This measurement should collapse the quantum state to an outcome that tells us what the $\mathrm{SUM}(x) \in \mathbb{Z}_k$ is. This process is described in figure 2.1.

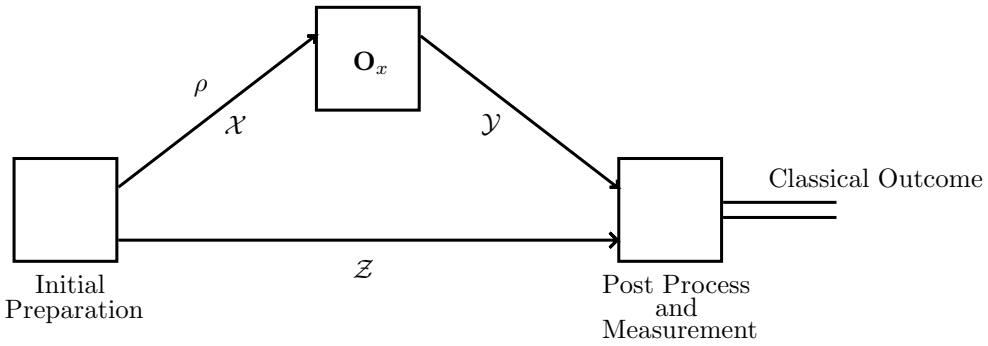

Figure 2.1: The general form of any quantum algorithm performing one query. $\mathcal{Z}$ is the workspace of the algorithm. $\mathcal{X}$ and $\mathcal{Y}$ are the input and output spaces of the oracle. ρ is part of the initial state sent to $\mathcal{O}_x$.

This is the form of *any* algorithm with one query. An *optimal* algorithm would consist of the *best* initial state and post-query computation such that measurement would give $\mathrm{SUM}(x)$ with highest probability of success. The problem of finding an optimal algorithm reduces to optimizing over the best possible initial state and post-query processing. These are the two key variables that our SDP will optimize over, for specific instances of n and k.

Dependent on them, we have another quantity, the probability of success. The SDP will be set up so that this probability of success is what is being maximized. However, since it is defined in terms of the initial state and post outcome measurement, those are really the two variables we are looking for that maximize the third.

To continue constructing our optimization problem, the oracle action and all constants must be clearly defined.

The oracle is a quantum channel as illustrated in figure 2.1. This channel is defined by an explicit operation on the input qubits. We denote this channel by $\mathcal{O}_x$ for some arbitrary string $x \in \mathbb{Z}_k^n$. If we consider the basis states $\{|a\rangle : a \in \mathbb{Z}_k^n\}$ as each representing an element of $\mathbb{Z}_k$, then we have that $X^a |b\rangle = |b + a\rangle$ where X^a is the generalized not operator on $\mathrm{L}(\mathbb{C}^k)$, raised to some power $a \in \mathbb{Z}_k$, and addition is done modulo k. Let the oracle $\mathcal{O}_x$ be a direct sum $(X^{a_1} \oplus \cdots \oplus X^{a_n})$ where each exponent a_i is the i-th digit of some string x. Then, application of $\mathcal{O}_x$ on the state $\frac{1}{\sqrt{n}} |0\rangle \oplus \cdots \oplus |0\rangle$ will produce $\frac{1}{\sqrt{n}} |a_1\rangle \oplus \cdots \oplus |a_n\rangle$. Furthermore, for $|i\rangle \otimes |b\rangle \in \mathbb{C}^n \otimes \mathbb{C}^k$, where $i \in \mathbb{Z}_n$ and $b \in \mathbb{Z}_k$ we get, $\mathcal{O}_x(|i\rangle \otimes |b\rangle) = |i\rangle \otimes |b + a_i\rangle$.

We will consider the Choi representation of such a channel which is denoted as J_x. Each quantum channel $\mathcal{O}_x$ can be represented by a unitary matrix U_x where each of the generalized not operators are the shift matrices on dimension k and basis states $|a\rangle$ are represented as basis vectors $e_a \in \mathbb{C}^k$. By definition $J_x = \mathrm{vec}\,(U_x)\,\mathrm{vec}\,(U_x)^*$. J_x will have a very simple eigenspectrum with only one nonzero eigenvalue. Its spectral decomposition is $\lambda u_x u_x^*$ where $\lambda = nk$ for all $x \in \mathbb{Z}_k^n$, where only the eigenvectors vary with respect to x.

Since $\mathcal{O}_x \in \mathrm{C}(\mathcal{X}, \mathcal{Y})$ where $\dim \mathcal{X} = \dim \mathcal{Y} = nk$, the corresponding Choi representation, J_x, is just a positive semidefinite operator in the space $\mathrm{L}(\mathcal{Y} \otimes \mathcal{X})$ which has dimension $(nk)^2$.

Essentially, what the algorithm is trying to do is discriminate from a set of predefined quantum channels via a state preparation and a measurement technique. We will refer to this strategy, consisting of the initial state and the measurement operators, as an *interactive measurement*. The collection of positive semidefinite operators used for measurement are what define the interactive measurement, with the condition that they all sum to $\mathbb{I} \otimes \rho$ for some $\rho \in \mathrm{D}(\mathcal{X})$ — which serves as the initial state for that interactive measurement.

Any quantum algorithm then is represented also as a collection of positive semidefinite operators that represent interactive measurement and live in the same space, $\{P_i\} \in \mathrm{Pos}(\mathcal{Y} \otimes \mathcal{X})$, where $i \in \mathbb{Z}_k$ and $\langle P_i, J_x \rangle$ is the overlap between the Choi representation of the oracle hiding some string x and P_i.

Consider some J_x that is the Choi representation of a unique string $x \in \mathbb{Z}_k^n$. Then

if $\mathrm{SUM}(x) = a \in \mathbb{Z}_k$, the overlap between P_a and J_x represents the probability that our interactive measurement succeeds in finding the correct answer for the sum of x. Call this overlap γ. The overlap of J_x and $\{P_i : i \neq \mathrm{SUM}(x)\}$ is the probability of getting an incorrect answer $(1 - \gamma)$. The problem then, is to find a collection of interactive measurements for which γ is as close to 1 as possible. If we want to find the *best* algorithm, we would like to maximize $\left\langle P_{\mathrm{SUM}(x)}, J_x \right\rangle$ *for all* x, arriving at the following set of linear constraints:

$$\forall x \in \mathbb{Z}_k^n : \left\langle J_x, P_{\mathrm{SUM}(x)} \right\rangle \geq \gamma$$

Additionally, a restriction on the form of each P_i must be placed to ensure that they are indeed quantum interactive measurements. More precisely, it must hold that $\sum_{i=1}^{k} P_i = \mathbb{I}_{\mathcal{Y}} \otimes \rho$ for some quantum state $\rho \in \mathrm{D}(\mathcal{X})$. This leads to the following optimization problem.

$$\begin{aligned} \text{maximize} \quad & \gamma \\ \text{subject to} \quad & \gamma \leq \left\langle P_{\mathrm{SUM}(x)}, J_x \right\rangle \text{ for all } x \in \mathbb{Z}_k^n \\ & \sum_{i=1}^{k} P_i = \mathbb{I}_{\mathcal{Y}} \otimes \rho \\ & \{P_i\} \in \mathrm{Pos}(\mathcal{Y} \otimes \mathcal{X}) \\ & \rho \in \mathrm{D}(\mathcal{X}) \end{aligned} \tag{4}$$

In this optimization problem, γ is the maximum probability of success by an interactive measurement algorithm, $\{P_i\}$ yields the actual quantum algorithm to achieve γ, and ρ is the initial state sent to the oracle as seen in figure 2.1. The aim is to maximize γ by finding some collection of positive semidefinite operators $\{P_i\}$ that each enjoy maximum overlap with strings of the same sum. Hence, the variables are the collection $\{P_i\}$ and ρ, while γ is directly dependent on them.

Feasible solutions to this SDP represent actual algorithms to solving the problem. We would also like to compute the dual to this optimization problem as it is more helpful for analysis of optimal probability of success for instances of n and k. In contrast with the primal, any solution to the dual will not lead to an algorithm, but it will give an upper bound to the probability of success for that particular instance. Just looking at the generic primal and dual in (1), it may not be apparent how exactly to derive the dual from the primal problem outlined above. We define some maps and super variables to help express it in a form similar to that of (1). Computation of the dual will then become more straightforward. This is done in the next section. Readers who wish to skip ahead to the

dual without seeing its step by step derivation or proof of strong duality may jump directly to section 2.1.3.

2.1.1 Computation of the One Query Dual

The primal optimization problem given in the previous section is already an SDP, although it may not be clear from its form. In this section we translate this primal optimization problem to the primal of the standard form for SDPs presented in (1). Once this is done, we will be able to use (1) as a guide to get the dual, and check for strong duality by applying Slater's Theorem for SDPs. If a reader is not interested in how exactly the dual problem is derived they can skip over this section to section 2.1.3 where the primal and dual for the single query case are shown together.

Let X be a matrix containing each variable on its diagonal as blocks.

$$X = \begin{pmatrix} \gamma & & & & \\ & P_0 & & & \\ & & \ddots & & \\ & & & P_{k-1} & \\ & & & & \rho \end{pmatrix} \tag{5}$$

X lives in the space $\mathrm{L}(\mathbb{C} \oplus (\mathcal{Y} \otimes \mathcal{X}) \oplus \ldots \oplus (\mathcal{Y} \otimes \mathcal{X}) \oplus \mathcal{X})$. This space will be denoted $\mathrm{L}(\mathcal{W})$. The dependent variable, γ, denotes the probability of success, and we want to bring that as close to 1 as possible. An objective function expressing this which follows the form in (1) is

$$\max \langle A, X \rangle$$

where

$$A = \begin{pmatrix} 1 & & & \\ & 0 & & \\ & & \ddots & \\ & & & 0 \end{pmatrix} \in \mathrm{L}(\mathcal{W}).$$

So far we have defined what form our variable X should have, and A from the triple (Φ, A, B) that defines an SDP in our standard form. Φ and B will be defined to ensure the constraints of the primal optimization problem (4) for single query quantum algorithms are met.

As outlined previously, any possible oracle representing a string in $\mathbb{Z}_k^n$ is represented as the Choi matrix of the quantum channel that defines its action. We label these matrices

uniquely as $J_1, \ldots, J_m$ where $m = k^n = |\mathbb{Z}_k^n|$. These are constants in our SDP and are used to specify the what function the SDP solves.

There are two groups of constraints in our optimization problem with different goals. The first group, defined by $\gamma \leq \left\langle P_{\mathrm{SUM}(x)}, J_x \right\rangle$ for all $x \in \mathbb{Z}_k^n$, in a sense defines the sum function. These constraints ensure that each interactive measurement outcome maps to the oracles that sum to the outcome it represents by forcing a maximum overlap between the two.

If instead of the sum function, we were interested in some different function $f : \mathbb{Z}_k^n \to \mathbb{Z}_k$ then those constraints would be rewritten as $\gamma \leq \left\langle P_{f(x)}, J_x \right\rangle$ for all $x \in \mathbb{Z}_k^n$.

Observe too, that the codomain of f or SUM dictates the size of our collection $\{P_i\}$ as each interactive measurements corresponds to a particular outcome of the function being considered.

The aim is to devise a map that enforces these constraints using the variable X which follows the standard SDP form shown in (1). Consider the map $\Phi : \mathrm{L}(\mathcal{W}) \to \mathrm{L}(\mathbb{C}^m)$, where

$$\Phi(X) = \begin{pmatrix} \gamma - \left\langle J_{x_1}, P_{\mathrm{SUM}(x_1)} \right\rangle & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \gamma - \left\langle J_{x_m}, P_{\mathrm{SUM}(x_m)} \right\rangle \end{pmatrix}. \tag{6}$$

Enforcing the first group of constraints in (4) then amounts to checking that

$$\Phi(X) \leq 0_m.$$

Next a similar map from our variable X defined in (5) is given with the aim of ensuring that each operator in the collection $\{P_i\}$ that we solve for composes a proper interactive measurement. For this to happen we have to implement the second constraint in (4) and check if $\{P_i\}$ sums to the identity on the output space $\mathcal{Y}$ tensored to some arbitrary density operator on the input space $\mathcal{X}$ of the oracle. Consider the map

$$\Psi(X) = \begin{pmatrix} \operatorname{Tr} \rho & 0 \\ 0 & \sum_{a \in \mathbb{Z}_k} P_a - \mathbb{I}_{\mathcal{Y}} \otimes \rho \end{pmatrix}.$$

Enforcing the constraint is then equivalent to checking

$$\Psi(X) = \begin{pmatrix} 1 & 0 \\ 0 & 0_{(nk)^2} \end{pmatrix}.$$

Lastly, enforcing the condition in the single query primal (4) that all $P_i \geq 0$ can be done by simply checking if $X \geq 0$. γ is a probability of success so restricting it to positive real values is satisfactory, and ρ is supposed to be a valid density operator which, in addition to having trace one, must be positive semi definite too.

To summarize, we have effectively translated the single query primal optimization problem in (4) to

$$\begin{aligned} &\text{maximize} && \langle A, X \rangle \\ &\text{subject to} && \Psi(X) = B \\ &&& \Phi(X) \leq C \\ &&& X \in \text{Pos}(\mathcal{W}) \end{aligned} \tag{7}$$

where $B = \begin{pmatrix} 1 & 0 \\ 0 & 0_{(nk)^2} \end{pmatrix} \in \text{L}(\mathbb{C} \oplus (\mathcal{Y} \otimes \mathcal{X}))$, $C = 0_m \in \text{L}(\mathbb{C}^m)$, and A, X, Φ and Ψ are defined as above. Let $\text{L}(\mathcal{B}) = \text{L}(\mathbb{C} \oplus (\mathcal{Y} \otimes \mathcal{X}))$ and $\text{L}(\mathcal{C}) = \text{L}(\mathbb{C}^m)$ for shorthand notation. This translated primal looks almost like the standard form in (1) except that there are two maps in the constraints, and one of them is an inequality. To get around this and arrive precisely at the form of (1), consider the map

$$\Xi\left(\begin{pmatrix} X & \cdot \\ \cdot & Z \end{pmatrix}\right) = \begin{pmatrix} \Psi(X) & 0 \\ 0 & \Phi(X) + Z \end{pmatrix}.$$

Here, $Z \in \text{L}(\mathcal{C})$ is a slack variable to help implement the inequality in (7). Using the super map Ξ, (7) is rewritten as

$$\begin{aligned} &\text{maximize} && \left\langle \begin{pmatrix} A & 0 \\ 0 & 0 \end{pmatrix}, \begin{pmatrix} X & 0 \\ 0 & Z \end{pmatrix} \right\rangle \\ &\text{subject to} && \Xi\left(\begin{pmatrix} X & 0 \\ 0 & Z \end{pmatrix}\right) = \begin{pmatrix} B & 0 \\ 0 & C \end{pmatrix} \\ &&& X \in \text{Pos}(\mathcal{W}) \\ &&& Z \in \text{Pos}(\mathcal{C}) \end{aligned}$$

which exactly mirrors the form of (1). Note that Z is unrestricted as long as it is positive semidefinite which is reflected by the objective function and the last constraint.

It is now clear how to derive the dual using (1) as a guide. Due to the standard form,

this dual must be

$$\begin{aligned} \text{minimize} \quad & \left\langle \begin{pmatrix} B & 0 \\ 0 & C \end{pmatrix}, \begin{pmatrix} Y & \cdot \\ \cdot & W \end{pmatrix} \right\rangle \\ \text{subject to} \quad & \Xi^* \left(\begin{pmatrix} Y & \cdot \\ \cdot & W \end{pmatrix} \right) \geq \begin{pmatrix} A & 0 \\ 0 & 0 \end{pmatrix} \\ & \begin{pmatrix} Y & \cdot \\ \cdot & W \end{pmatrix} \in \text{Herm}\left(\mathcal{B} \oplus \mathcal{C}\right) \end{aligned}$$

where Y and W are the dual variables, A, B, and C are as defined previously, and Ξ^* is the dual map of Ξ. The main goal of this section is to arrive at a dual of the single query primal in (4) that provides the same readability and further insight into the problem. Therefore, we want to translate the dual above back into terms involving our original problem. Most of the effort will revolve around in finding the action of Ξ^*. This action is fixed because Ξ^* must satisfy the following equation

$$\langle Y, \Xi\left(X\right)\rangle = \langle \Xi^*\left(Y\right), X\rangle\,.$$

This equation must hold for any completely positive, trace preserving map. It is straightforward to see that Ξ^* is uniquely given by

$$\Xi^* \left(\begin{pmatrix} Y & \cdot \\ \cdot & W \end{pmatrix} \right) = \begin{pmatrix} \Psi^*(Y) + \Phi^*(W) & 0 \\ 0 & W \end{pmatrix}.$$

Substituting this into the dual gives

$$\begin{aligned} \text{minimize} \quad & \left\langle \begin{pmatrix} B & 0 \\ 0 & C \end{pmatrix}, \begin{pmatrix} Y & \cdot \\ \cdot & W \end{pmatrix} \right\rangle \\ \text{subject to} \quad & \begin{pmatrix} \Psi^*(Y) + \Phi^*(W) & 0 \\ 0 & W \end{pmatrix} \geq \begin{pmatrix} A & 0 \\ 0 & 0 \end{pmatrix} \\ & \begin{pmatrix} Y & \cdot \\ \cdot & W \end{pmatrix} \in \text{Herm}\left(\mathcal{B} \oplus \mathcal{C}\right) \end{aligned}$$

which is equivalent to

$$\begin{aligned} \text{minimize} \quad & \langle B, Y\rangle + \langle C, W\rangle \\ \text{subject to} \quad & \Psi^*(Y) + \Phi^*(W) \geq A \\ & Y \in \text{Herm}(\mathcal{B}) \\ & W \in \text{Pos}(\mathcal{C}). \end{aligned}$$

Here, C is the zero matrix on $\mathrm{L}(\mathcal{C})$ so we can drop the second term of the objective function to arrive at

$$\begin{array}{ll} \text{minimize} & \langle B, Y\rangle \\ \text{subject to} & \Psi^*(Y) + \Phi^*(W) \geq A \\ & Y \in \mathrm{Herm}(\mathcal{B}) \\ & W \in \mathrm{Pos}(\mathcal{C}). \end{array} \tag{8}$$

What remains is to provide definitions of Ψ^* and Φ^*, substitute them in and simplify to get our desired dual.

Finding Ψ^* Recalling the definition of $\Psi(X)$. Let $Y \in \mathrm{L}(\mathcal{B})$ be expressed by $\begin{pmatrix} y_1 & \cdot \\ \cdot & Y_2 \end{pmatrix}$, where $y_1 \in \mathrm{L}(\mathbb{C})$ and $Y_2 \in \mathrm{L}(\mathcal{X} \otimes \mathcal{Y})$. Using the relation

$$\langle Y, \Psi(X)\rangle = \langle \Psi^*(Y), X\rangle$$

we get that

$$\Psi^*\left(\begin{pmatrix} y_1 & \cdot \\ \cdot & Y_2 \end{pmatrix}\right) = \begin{pmatrix} 0 & & \cdots & 0 \\ & Y_2 & & \\ \vdots & & \ddots & \vdots \\ & & Y_2 & \\ 0 & & \cdots & y_1\mathbb{I}_{\mathcal{X}} - \mathrm{Tr}_{\mathcal{Y}}(Y_2) \end{pmatrix} \in \mathrm{L}(\mathcal{W}).$$

Proof: For any $Y \in \mathrm{L}(\mathcal{B})$ and $X \in \mathrm{L}(\mathcal{W})$ we have

$$\begin{aligned}
\langle Y, \Psi(X)\rangle &= \left\langle \begin{pmatrix} y_1 & \cdot \\ \cdot & Y_2 \end{pmatrix}, \begin{pmatrix} \operatorname{Tr}\rho & 0 \\ 0 & \sum_{a\in\mathbb{Z}_k} P_a - \mathbb{I}_{\mathcal{Y}} \otimes \rho \end{pmatrix} \right\rangle \\
&= \langle y_1, \operatorname{Tr}(\rho)\rangle + \left\langle Y_2, \sum_{a\in\mathbb{Z}_k} P_a - \mathbb{I}_{\mathcal{Y}} \otimes \rho \right\rangle \\
&= \langle y_1 \mathbb{I}_{\mathcal{X}}, \rho\rangle + \left\langle Y_2, \sum_{a\in\mathbb{Z}_k} P_a \right\rangle - \langle Y_2, \mathbb{I}_{\mathcal{Y}} \otimes \rho\rangle \\
&= \langle y_1 \mathbb{I}_{\mathcal{X}}, \rho\rangle + \sum_{a\in\mathbb{Z}_k} \langle Y_2, P_a\rangle - \langle Y_2, \mathbb{I}_{\mathcal{Y}} \otimes \rho\rangle \\
&= \langle y_1 \mathbb{I}_{\mathcal{X}}, \rho\rangle + \sum_{a\in\mathbb{Z}_k} \langle Y_2, P_a\rangle - \operatorname{Tr}(Y_2^* (\mathbb{I}_{\mathcal{Y}} \otimes \rho)) \\
&= \langle y_1 \mathbb{I}_{\mathcal{X}}, \rho\rangle + \sum_{a\in\mathbb{Z}_k} \langle Y_2, P_a\rangle - \langle \operatorname{Tr}_y Y_2, \rho\rangle \\
&= \left\langle \begin{pmatrix} 0 & & \cdots & 0 \\ & Y_2 & & \\ \vdots & & \ddots & \vdots \\ & & Y_2 & \\ 0 & & \cdots & y_1 \mathbb{I}_{\mathcal{X}} - \operatorname{Tr}_{\mathcal{Y}}(Y_2) \end{pmatrix}, \begin{pmatrix} \lambda & & & & \\ & P_0 & & & \\ & & \ddots & & \\ & & & P_{k-1} & \\ & & & & \rho \end{pmatrix} \right\rangle \\
&= \langle \Psi^*(Y), X\rangle .
\end{aligned}$$

□

Finding Φ^* Φ^* is a bit more complicated to specify. Recalling the definition of $\Phi(X)$ we have $m = k^n$ rows, where on each row, we have an expression with a unique J_x for *all* $x \in \mathbb{Z}_k^n$. These Choi matrix representations of the strings with length n and digit size k are paired up along with the interactive measurement operator P_i for which $i = \mathrm{SUM}(x) \in \mathbb{Z}_k$. There are k^n of these constraints, one for every string which is possibly given. They are partitioned evenly into k sum classes. A *sum class* is a set of all strings in $\mathbb{Z}_k^n$ that have equal sum. Since the sum function partitions the space of all strings evenly, each of the k sum classes has k^{n-1} elements. A sum class in which all strings have the sum s is denoted ζ_s. Remember the form of Φ defined in (6). This map is used in the primal to enforce

the constraints that the value of *every* $\left\langle J_x, P_{\mathrm{SUM}(x)} \right\rangle$ is greater than γ - the probability of success. Φ orders these constraints along the diagonal of a matrix with k^n elements. There is no significance to the order of which constraint comes first, as they are all checked to be less than or equal to zero. Therefore, the action of Φ is equivalent up to permutation of the ordering on the diagonal of its output. This notion plays prominently in the proof below. The dual $\Phi^* : \mathrm{L}(\mathcal{C}) \to \mathrm{L}(\mathcal{W})$ is

$$\Phi^*(W) = \begin{pmatrix} \sum_{i\in[m]} w_i & & & \cdots & 0 \\ & -\left(\sum_{x\in\zeta_0} w_{i(x)} J_x\right) & & & \\ \vdots & & \ddots & & \vdots \\ & & & -\left(\sum_{x\in\zeta_{k-1}} w_{i(x)} J_x\right) & \\ 0 & \cdots & & & 0 \end{pmatrix}.$$

Here, w_i corresponds to each of the diagonal elements of the matrix $\Phi^*(X) = W \in \mathrm{L}(\mathbb{C}^m)$ indexed by some i in $[m]$. These elements are all non negative real numbers as W in (8) must be hermitian. Note that there are m distinct strings in $\mathbb{Z}_k^n$ and m distinct diagonal entries of W. In Φ^* each diagonal entry of W is paired uniquely to some J_{x_i} for $x_i \in \mathbb{Z}_k^n$ via the function $i(x)$. This function is just a mapping from the set of strings in $\mathbb{Z}_k^n$ to an indexing set that is also used to index the diagonal entries of W. There will be more said on the significance of the w_i values when we arrive at the final form of the dual. A proof of the correctness of Φ^* as specified above follows.

Proof: The validity of Φ^* amounts to verifying that it satisfies

$$\langle W, \Phi(X) \rangle = \langle \Phi^*(X), W \rangle$$

and can be verified as follows.

$$\langle W, \Phi(X)\rangle = \left\langle \begin{pmatrix} w_1 & & \\ & \ddots & \\ & & w_m \end{pmatrix}, \begin{pmatrix} \gamma - \langle J_{x_1}, P_{SUM(x_1)}\rangle & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \gamma - \langle J_{x_m}, P_{SUM(x_m)}\rangle \end{pmatrix} \right\rangle$$

$$= \sum_{x\in\mathbb{Z}_k^n} w_{i(x)}\gamma - w_{i(x)}\langle J_x, P_{SUM(x)}\rangle$$

$$= \left\langle \begin{pmatrix} \sum_{i\in[m]} w_i & & & \cdots & 0 \\ & -\left(\sum_{x\in\zeta_0} w_{i(x)}J_x\right) & & & \\ \vdots & & \ddots & & \vdots \\ & & & -\left(\sum_{x\in\zeta_{k-1}} w_{i(x)}J_x\right) & \\ 0 & \cdots & & & 0 \end{pmatrix}, \begin{pmatrix} \gamma & & & \\ & P_0 & & \\ & & \ddots & \\ & & & P_{k-1} \\ & & & & \rho \end{pmatrix} \right\rangle$$

$$= \langle \Phi^*(W), X\rangle .$$

□

Now that Φ^* and Ψ^* have been computed, we can rewrite the dual in (8) in a more expressive form. Since we can write Y as $\begin{pmatrix} y_1 & \cdot \\ \cdot & Y_2 \end{pmatrix}$ for $y_1, Y_2 \geq 0$, the objective function becomes

$$\text{minimize } \left\langle \begin{pmatrix} 1 & 0 \\ 0 & 0_{(nk)^2} \end{pmatrix}, \begin{pmatrix} y_1 & \cdot \\ \cdot & Y_2 \end{pmatrix} \right\rangle \Leftrightarrow \text{minimize } y_1.$$

Plugging in Φ^* and Ψ^* into the inequality

$$\Psi^*(Y) + \Phi^*(W) \geq A$$

gives

$$\begin{pmatrix} \sum_{i\in[m]} w_i & & & \cdots & 0 \\ & Y_2 - \left(\sum_{x\in\zeta_0} w_{i(x)} J_x\right) & & & \\ \vdots & & \ddots & & \vdots \\ & & & Y_2 - \left(\sum_{x\in\zeta_{k-1}} w_{i(x)} J_x\right) & \\ 0 & & & \cdots & y_1 \mathbb{I}_{\mathcal{X}} - \dim(\mathcal{Y}) \operatorname{Tr}_{\mathcal{Y}}(Y_2) \end{pmatrix}$$
$$\geq \begin{pmatrix} 1 & & \cdots & 0 \\ & 0 & & \\ \vdots & \ddots & & \vdots \\ & & 0 & \\ 0 & & \cdots & 0 \end{pmatrix}.$$

Compiling these constraints and objective, we get the dual to be

$$\begin{aligned} \text{minimize} \quad & y_1 \\ \text{subject to} \quad & \sum_{i\in[m]} w_i \geq 1 \\ & Y_2 \geq \left(\sum_{x\in\zeta_s} w_{i(x)} J_x\right) \quad \forall s \in \mathbb{Z}_k \\ & y_1 \mathbb{I}_{\mathcal{X}} \geq \dim(\mathcal{Y}) \operatorname{Tr}_{\mathcal{Y}}(Y_2) \end{aligned} \tag{9}$$

where

$$\begin{aligned} & y_1 \in \mathbb{R} \\ & Y_2 \in \operatorname{Herm}(\mathcal{Y} \otimes \mathcal{X}) \\ & w_i \geq 0. \end{aligned}$$

This concludes the specification of our primal and dual for the SDP that optimizes the probability of success of any algorithm attempting to solve the SUM mod k problem on

strings of length n with one query. For now, we just leave the reader without discussion as to what this SDP tells us about the problem and what exactly the primal and dual mean. Next, we turn our attention to showing strong duality exists by Slater's theorem for SDPs. In section 2.1.3 we will analyze the primal and dual from this section to gain insight into the sum problem with one query. In section 2.2.2 a closed form for the optimal probability of success will be shown.

2.1.2 Strong Duality

All SDPs have a property called weak duality. This means that the primal optimal value, α is always less than or equal to β, the dual optimal value. Weak duality is nice as it implies that any feasible solution of the dual gives an upper bound to the primal optimal value, while any feasible solution to the primal gives a lower bound on the dual optimal value. Weak duality is useful because the goal of any optimization problem is usually to find or at least shed some insight as to what its optimal value is. Solving for that optimal value is often computationally very difficult. Using Slater's theorem we can show that α equals β and that both values are achievable by corresponding feasible solutions. Having no gap between the optimal values of the primal and dual of some optimization problem is very useful because it promises that examining either will be just as fruitful to gaining insight about their optima. In our case, we are chiefly interested in the optimal value of the primal (4). It corresponds to the optimal probability of success for any quantum algorithm which makes one query to a black box representing an arbitrary string of fixed length and digit size in attempt to compute the string's sum. The remainder of this section shows that Slater's conditions 1 and 2 hold for this SDP. A note to the reader: strong duality is not used in finding the exact probability of success bound for computing the sum function of a hidden string. The proof of this bound simply finds a dual feasible solution and shows that the objective value of the dual equals the objective value achieved by the algorithm in [13]. Therefore this section is not a prerequisite for that result. Knowing that this SDP exhibits strong duality was a stepping stone along the way to come up with the proof however, since it implies that a construction of a feasible dual solution that matches the optimal primal value is possible.

Claim: For the SDP given by primal (4) and dual (9) $\alpha = \beta$ and both are achieved by respective feasible solutions.

Proof: To show that strong duality exists and both optima are achievable, both Slater conditions must be met. This amounts to showing that the dual and primal are both

strictly feasible. From (3), this means we must show

1. $\exists\ Y\ \mathrm{Herm}(\mathcal{Y})$ where $\Phi^*(Y) > A$
2. $\exists X > 0$ where $\Phi(X) = B$

Slater's theorem (3) corresponds to the general SDP form given in section 1.2. To show it also holds for the sum function SDP for one query algorithms, we first substitute our definitions for the primal and dual variable and maps. Substituting in the appropriate definitions the two conditions are re-expressed as

1. $\exists \begin{pmatrix} Y & \\ & W \end{pmatrix} \in \mathrm{Herm}(\mathbb{C} \oplus (\mathcal{Y} \otimes \mathcal{X}) \oplus \mathbb{C}^m)$ such that
   $\Psi^*(Y) + \Phi^*(W) > A$ where $Y = \begin{pmatrix} y_1 & \\ & Y_2 \end{pmatrix}$
2. $\exists X \in \mathrm{Pd}(\mathbb{C} \oplus (\mathcal{Y} \otimes \mathcal{X})^k \oplus \mathcal{X}),\ Z \in \mathrm{Pd}(\mathbb{C}^m)$ such that
   $\Psi(X) = \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix}$ and $\Phi(X) + Z = 0$

1. Strict Dual Feasibility: In the dual, the variable Y is composed of y_1 and Y_2. Let $\delta > 0$, $Y_2 = (nm + \delta)\mathbb{I}_{\mathcal{Y}\otimes\mathcal{X}}$ and $y_1 > (nm + \delta)\dim(\mathcal{Y})$ where $m = k^n$. Also, let $W = \mathbb{I}_{\mathbb{C}^m}$. According to the action of Ψ^* and Φ^* we get that

$$\begin{pmatrix} 0 & & & & \\ & (nm+\delta)\mathbb{I}_{\mathcal{Y}\otimes\mathcal{X}} & & & \\ & & \ddots & & \\ & & & (nm+\delta)\mathbb{I}_{\mathcal{Y}\otimes\mathcal{X}} & \\ & & & & y_1\mathbb{I}_{\mathcal{X}} - \mathrm{Tr}(\mathcal{Y}_2) \end{pmatrix} + \begin{pmatrix} \mathrm{Tr}(W) & & & & \\ & -\sum_{x\in\zeta_0} J_x & & & \\ & & \ddots & & \\ & & & -\sum_{x\in\zeta_{k-1}} J_x & \\ & & & & 0 \end{pmatrix}$$

must be strictly greater than

$$\begin{pmatrix} 1 & & & \\ & 0 & & \\ & & \ddots & \\ & & & 0 \end{pmatrix}.$$

It is easy to check that this is true. The trace of W is $m > 1$. We know that $\lambda_1(J_x) = nk$ which implies

$$\lambda_1 \left(\sum_{x \in \zeta_s} J_x \right) \leq (nk)k^{n-1} = nm.$$

Since $\delta > 0$, it must hold that

$$nm + \delta)\mathbb{I}_{\mathcal{Y} \otimes \mathcal{X}} - \sum_{x \in \zeta_s} J_x > 0$$

for all $i \in \mathbb{Z}_k$. Finally,

$$y_1 \mathbb{I}_{\mathcal{X}} - \mathrm{Tr}(\mathcal{Y}_2) = (y_1 - \dim(\mathcal{Y})(nm + \delta))\mathbb{I}_{\mathcal{X}} > 0$$

because $y_1 > \dim(\mathcal{Y})(nm + \delta)$.

2. Strict Primal Feasibility: We need to give any positive definite X and Z such that

$$(X) = \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix} \text{ and } \Phi(X) + Z = 0.$$

Remember that X is comprised of the direct sum of the primal variables γ, ρ, $P_0, \ldots, P_{k-1}$. These must all be positive definite due to the second condition. ρ should be a density matrix so let $\rho = \frac{1}{\dim(\mathcal{X})}\mathbb{I}_{\mathcal{X}}$. Let $P_0, \ldots, P_{k-1} = \frac{1}{k}\mathbb{I}_{\mathcal{X}} \otimes \rho$ and $\gamma = 0$. It holds that

$$x \in \mathbb{Z}_k^n \left\langle J_x, \frac{1}{k}\mathbb{I}_{\mathcal{X}} \otimes \rho \right\rangle > 0$$

because $J_x = \lambda u_x u_x^*$ for $\lambda = nk$ and $u_x \in \mathcal{S}(\mathcal{X})$. The condition that

$$\Psi(X) = \begin{pmatrix} 1 & 0 \\ 0 & 0_{(nk)^2} \end{pmatrix}$$

is satisfied since

$$\mathrm{Tr}\, \rho = 1$$

and

$$\sum_{a\in\mathbb{Z}_k} P_a - \mathbb{I}_{\mathcal{Y}} \otimes \rho = 0.$$

If we let $Z = \bigoplus_{x\in\mathbb{Z}_k^n} \left(\left\langle J_x, \frac{1}{k}\mathbb{I}_{\mathcal{X}} \otimes \rho\right\rangle\right)$ where $\bigoplus$ is the direct sum over 1 by 1 matrices, then $Z \geq 0$ and $\Phi(X) + Z = 0$ is also satisfied because

$$\begin{aligned}
\Phi(X) + Z &= \bigoplus_{x\in\mathbb{Z}_k^n} \left(\gamma - \left\langle J_x, P_{\mathrm{SUM}(x)}\right\rangle\right) + \bigoplus_{x\in\mathbb{Z}_k^n} \left(\left\langle J_x, \frac{1}{k}\mathbb{I}_{\mathcal{X}} \otimes \rho\right\rangle\right) \\
&= \bigoplus_{x\in\mathbb{Z}_k^n} \left(\gamma - \left\langle J_x, \frac{1}{k}\mathbb{I}_{\mathcal{X}} \otimes \rho\right\rangle + \left\langle J_x, \frac{1}{k}\mathbb{I}_{\mathcal{X}} \otimes \rho\right\rangle\right) \\
&= \bigoplus \gamma = \begin{pmatrix} 0 & & \\ & \ddots & \\ & & 0 \end{pmatrix}.
\end{aligned}$$

□

2.1.3 Summary

So far the SDP for the problem of computing the sum of a hidden string with a single query to the oracle has been specified. Both the primal and the dual have been given and have been shown to exhibit strong duality. In this section we take a step back and discuss what exactly they mean and what we can hope to gain through this optimization problem.

The primal

maximize	γ
subject to	$\gamma \leq \left\langle P_{\mathrm{SUM}(x)}, J_x\right\rangle$ for all $x \in \mathbb{Z}_k^n$
	$\sum_{i=1}^{k} P_i = \mathbb{I}_{\mathcal{Y}} \otimes \rho$
	$\{P_i\} \in \mathrm{Pos}(\mathcal{X})$

The primal constitutes of optimizing over the following variables.

ρ – The input state to the oracle shown in figure (2.1).

$\{P_0, \ldots, P_{k-1}\}$ – A collection of positive semidefinite operators that constitute an interactive measurement. This collection of operators, along with the initial state in effect constitute any possible quantum algorithm that can be implemented for this problem.

γ – The probability of success of a feasible set of $\{\mathcal{P}_a\}$ and initial state ρ successfully identifying the correct oracle's sum.

Here, γ is bound by the success of a protective measurement's ability successfully discern which sum class ζ_s the hidden string is a member of. This ability is given by the correct semidefinite operator of the interactive measurement having maximum overlap with all strings in a sum class. In such a case, that measurement is most likely to occur when the interactive measurement is applied to the hidden string's Choi representation. This Choi representation lives in the space $\mathrm{L}(\mathcal{Y} \otimes \mathcal{X})$ so it has dimension $(nk)^2$. The interactive measurement also has this dimension. On the other hand, each sum class has k^n elements. This grows faster than $(nk)^2$, so the larger k^n gets the harder it would be to find k positive semidefinite operators that do not have a large overlap with one another. The larger the overlap between the P_a operators, the less the probability of success will be.

The constants are

k – The string digit alphabet size.

n – The string length of the hidden string.

J_x – The Choi representation of the oracle hiding some string in $\mathbb{Z}_k^n$
There are k^n J_x's

Each n and k constitute a different instance of the problem. However, due to the uselessness theorem from [12], we need not concern ourselves with values of n greater than 2.

Any collection $\mathcal{P}_a$ and ρ that are feasible also give a lower bound on α the maximum probability of success possible. Therefore if an initial state and interactive measurement are found that give a better probability of success than the algorithm in [13], it would constitute a proof that it is non-optimal. On the other hand if the algorithm matches α for arbitrary n and k we know the it is optimal in the one query case. This logic extends to our SDP for more than a single query which will be given in section 3.1.

The Dual

$$\begin{aligned} &\text{minimize} && y_1 \\ &\text{subject to} && \sum_{i\in[m]} w_i \geq 1 \\ & && Y_2 \geq \left(\sum_{x\in\zeta_s} w_{i(x)} J_x\right) \quad \forall s \in \mathbb{Z}_k \\ & && y_1 \mathbb{I}_{\mathcal{X}} \geq \dim(\mathcal{Y}) \operatorname{Tr}_{\mathcal{Y}}(Y_2) \end{aligned} \tag{9}$$

The dual constitutes of optimizing over the following variables.

Y_2 – Some positive semidefinite operator that must be greater than or equal to the sum of all the weighted Choi representations of strings in the same sum class.

W – A diagonal matrix of dimention k^n. The entries along its diagonal are labeled w_i. Each w_i is the weight of some hidden string Choi representation in the sum $\sum_{x\in\zeta_k} w_{i(x)} J_x$. $i(x)$ here is a function uniquely mapping each string x to some index of the diagonal elements of W.

y_1 – a real number bounded below by $\frac{1}{\dim(\mathcal{X})} \operatorname{Tr} Y_2$.

where the constants are the same as in the primal problem. Recall that n and k define a set of all possible strings that are candidates for the one that is hidden. We can imagine an adversary that can have some sort of selection strategy which governs what string we are given. This selection strategy induces a prior probability distribution over the set of all possible strings defined by n and k. The weights that exist along the diagonal of the variable W are precisely that. They must all be non negative real numbers since W must be positive semidefinite and they must all sum to something less than or equal to one. The diagonal of W is therefore the probability vector which the adversary can fiddle with to decrease y_1 as much as possible. Since we know from the previous section that this SDP has strong duality. The optimum value for y_1 is equal to the optimal value for γ in the primal, and both are achievable. y_1 then, for an arbitrary feasible solution Y_2 and

W is an upper bound on the optimal probability of success. Any solution to the dual that yields a value for y_i that is less than 1 gives a very interesting result: No quantum algorithm can solve the sum problem with one query with probability of success greater than y_1. This dual is more computationally difficult to solve than the primal. However, particularly for the multiple query scenario, a simplification can be made. The variable W, the adversary's induced prior, can be taken to equal $\frac{1}{m}\mathbb{I}_{\mathbb{C}^m}$. This restricts the problem to finding the smallest y_1 where the quantum algorithm is promised that the hidden string is chosen from a uniform distribution over all possible strings.

In the next section results involving the one query case are discussed.

2.2 Single Query SDP Results

In this section we present results attained from the single query SDP. It turns out that the Meyer and Pommersheim algorithm is optimal for the single query case. First, numeric results are presented for values of k that are tractable. Then analysis resulting in the closed form for the probability of success for arbitrary k is presented.

The main result in this section tells us that the optimal probability of success of quantum algorithms computing the sum of strings with length $n = 2$ and arbitrary alphabet sizes is $\frac{2}{k}$. Since $2 \leq \sqrt{k}$ for $k > 3$ applying the optimal single query quantum algorithm twice for $n = 4$ when $k > 3$ is no better than guessing. For $k = 3$ we can apply the optimal quantum algorithm twice and get the sum of a 4 digit string with probability of success equal to $\frac{4}{9}$ which is better than guessing, but worse than two sequential queries which the general algorithm from [13] makes to achieve probability of success $\frac{2}{3}$.

2.2.1 Numerical Results

To get an idea about what happens with the optimal probability of success γ and see how the single query SDP behaves for specific cases, numerical solutions will be presented for a few tractable cases of k. These results were obtained by running a linear solver for SDPs, CVX, in MATLAB on the Feynman supercompluting cluster at the Institute for Quantum Computing in Waterloo. Each result of running the primal for specific values of n and k provides not only a data point for α - the optimal probability of success for any single query quantum algorithm, but also provides the optimal algorithm from that case. Figure 2.2 shows a plot of different optimal values for $n = 2$ and k ranging from 2 to 7 - the

largest tractable case for the hardware at hand. These values coincide with the probability of success achieved by the Meyer and Pommersheim algorithm.

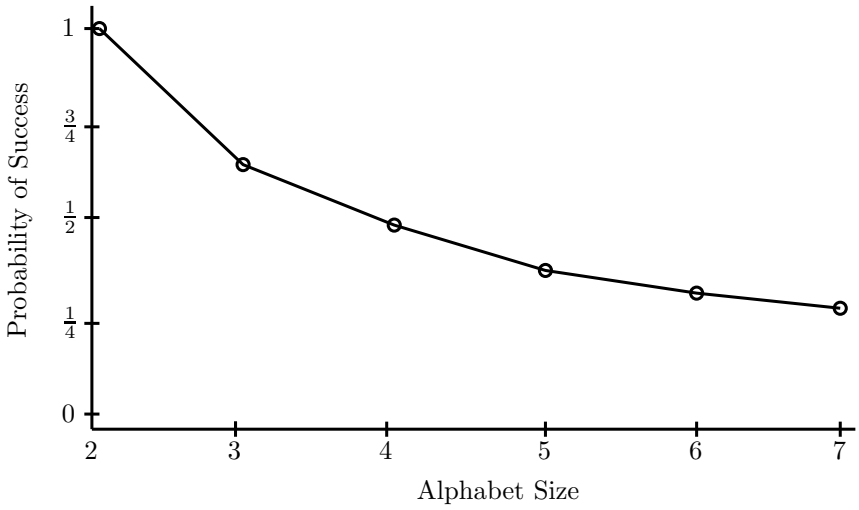

Figure 2.2: A plot of numerically computed optima of the primal (and dual) for the one query SDP and $n = 2$.

Tractability Running the solver on instances of the problem for increasing values of n and k quickly becomes intractable. The main reason is that k quickly increases the dimension of the matrices that optimized over. The primal objective function tries to maximize the overlap between X and A. This reduces to getting γ as close to one as possible, but in the constraints we are still optimizing over the best collection of $\{P_a\}$. The dimension of each of these operators is governed by the dimension of the Choi representation of the oracle which hides a string. For $x \in \mathbb{Z}_k^n$, $\dim(J_x) = (nk)^2$. n is fixed to two, so this dimension grows in terms of k as: $4k^2$. When $k = 8$, $\dim(J_x) = 256$. As the size of the variables P_a increases, the memory required to perform matrix multiplication increases. Hence, memory is the primary bottleneck during computation. Regardless of this computational limitation, it was possible to get enough data points to see a trend and perform analysis for the single query case. This issue becomes severely crippling in the two query case however, as even for the most trivial cases large nodes of the supercomputing cluster ran out of memory (128GB). The dimension of the matrices being optimized over increases with respect to the number of queries made as $(nk)^{2q}$. The reason for this will become apparent when the multi-query SDP is shown in Chapter 3.

The few data points shown in figure 2.2 however, are invaluable to our insight into the single query problem, as they play a central role in the analytic proof of optimality of the algorithm. This is attributed to the fact that along with the optima displayed in figure 2.2, the numeric solutions also provide concrete values for each of the other variables of the primal and dual.

2.2.2 Analytic Results

The major result from the single query SDP is that for adversarially chosen strings from $\mathbb{Z}_k^n$ the optimal probability of success of any quantum algorithm making one query is $\frac{2}{k}$. This upper bound matches the lower bound of the algorithm in [13] which proves that algorithm's optimality for this case.

The single query SDP has strong duality so we can look toward either the optimal value of the primal or dual with equal impact toward a proof. These optima are bounded above or below respectively by the other variables in the problem.

The main variable of the dual is Y_2. Once a feasible Y_2 is found, it then bounds y_1 below. Therefore, y_1 is considered as the value of the dual objective function. Any Y_2 which minimizes y_1 according to the dual constraints cannot push y_1 to be lower than the bound provided by the algorithm due to weak duality. If some Y_2 is given which places the bound of y_1 at the probability of success of the algorithm, then we must conclude that this value of y_1 is the dual optimum and that the algorithm is optimal. Strong duality and strict dual and primal feasibility suggest that there does exist some algorithm and, Y_2 which cause the primal and dual optimal to agree. The approach which lead to the proof of optimality began by trying to find some Y_2 for which $y_1 = \frac{2}{k}$ for arbitrary k. Examination of dual feasible solutions was preferential because any feasible solution which bounds y_1 somewhere less than 1 would provide at least an interesting upper bound for the probability of success. On the other hand, if primal solutions were examined, we would have to only consider those that give a primal objective function value greater than that of the current algorithm. Finding one would be nice because primal feasible solutions are actually interactive measurements that give algorithms for achieving the objective function value they yield. However, there is no guarantee that such solutions exist if the Meyer and Pommersheim algorithm is optimal.

Due to the constraint $y_1 \mathbb{I}_{\mathcal{X}} \geq \mathrm{Tr}_{\mathcal{Y}} Y_2$ it is clear that y_1, the objective value of the dual, is bounded below by the spectral norm of $\mathrm{Tr}_{\mathcal{Y}} Y_2$. A valid proof of some value y_1 that is less than 1 (non trivial), would constitute showing that there always exists a Y_2 whose spectral norm when tracing over the output space of the oracle is that value. There is

also the variable $W \in \mathrm{Pos}(\mathbb{C}^M)$ which must satisfy $\mathrm{Tr}(W) \geq 1$. The diagonal entries of this variable denote the prior probability distribution over strings $x \in \mathbb{Z}_k^n$. We show that the proof works for a uniform distribution over the space of strings. Since the value of y_1 achieved by setting $W = \frac{1}{m}\mathbb{I}_{\mathbb{C}^m}$ will be shown to be optimal, it means that an adversary cannot do better than providing the algorithm with strings uniformly at random for $n = 2$, arbitrary finite k, and $q = 1$.

Theorem: For $n = 2$ and any finite $k \geq 2$, there always exists a $Y_2 \in \mathrm{Herm}(\mathcal{Y} \otimes \mathcal{X})$ with $||\mathrm{Tr}_{\mathcal{Y}} Y_2||_\infty = \frac{2}{k}$ which satisfies

$$Y_2 \geq \frac{1}{m} \sum_{x \in \zeta_s} J_x$$

for all $s \in \mathbb{Z}_k^n$, and SUM equivalence classes $\zeta_s = \{x \in \mathbb{Z}_k^n : \mathrm{SUM}(x) = s\}$, where $\mathcal{X}$ and $\mathcal{Y}$ are the input and output complex Euclidean vector spaces that an oracle O_x acts on as shown in figure 2.1, and $\frac{1}{m}$ is the probability distribution on the prior likelihood of the oracle hiding any string.

Proof: The proof sketch is as follows.

1. Let $A_s = \sum_{x \in \zeta_s} J_x$. Examine the structure of each A_s.
2. Give an explicit construction of $Y_2 \in \mathrm{Herm}(\mathcal{Y} \otimes \mathcal{X})$ such that $Y_2 \geq \frac{1}{m} A_s$ holds for all $s \in \mathbb{Z}_k$ and arbitrary $k > 2$.
3. Show that $||\mathrm{Tr}_{\mathcal{Y}} Y_2||_\infty = \frac{2}{k}$.

The structure of A_s: Each J_x is the Choi representation of the respective oracle operator O_x with unitary representation U_x for some string $x \in \mathbb{Z}_k^n$. In this case, n is fixed to be 2 so we take $x = (a_0, a_1)$. Therefore we can express A_s as

$$A_s = \sum_{x \in \zeta_s} J_x = \sum_{x \in \zeta_s} \mathrm{vec}\,(U_x)\,\mathrm{vec}\,(U_x)^*. \tag{10}$$

But U_x has form

$$\begin{pmatrix} X^{a_0} & 0_k \\ 0_k & X^{a_1} \end{pmatrix}$$

where $a_0, a_1 \in \mathbb{Z}_k$ are the digits of the hidden string, X is the generalized-not operator over $\mathrm{L}(\mathbb{C}^k)$, and 0_k is the all zeros matrix. In light of this, we can deconstruct $\mathrm{vec}\,(U_x)$ as $\mathrm{vec}\,(X^{a_0} \oplus X^{a_1}) = \mathrm{vec}\,(X^{a_0} \oplus 0_k) + \mathrm{vec}\,(0_k \oplus X^{a_1})$. Then we get that each J_x can be written as

$$\begin{aligned} J_x &= \left(\mathrm{vec}\,(X^{a_0} \oplus 0_k) + \mathrm{vec}\,(0_k \oplus X^{a_1})\right)\left(\mathrm{vec}\,(X^{a_0} \oplus 0_k) + \mathrm{vec}\,(0_k \oplus X^{a_1})\right)^* \\ &= \mathrm{vec}\,(X^{a_0} \oplus 0_k)\,\mathrm{vec}\,(X^{a_0} \oplus 0_k)^* \\ &\quad + \mathrm{vec}\,(X^{a_0} \oplus 0_k)\,\mathrm{vec}\,(0_k \oplus X^{a_1})^* \\ &\quad + \mathrm{vec}\,(0_k \oplus X^{a_1})\,\mathrm{vec}\,(X^{a_0} \oplus 0_k)^* \\ &\quad + \mathrm{vec}\,(0 \oplus X^{a_1})\,\mathrm{vec}\,(0_k \oplus X^{a_1})^*. \end{aligned} \tag{11}$$

Next, observe that for some matrix M of dimension k, we have

$$\mathrm{vec}\,(M \oplus 0) = \sum_{l=0}^{k-1} e_0 \otimes e_l \otimes e_0 \otimes c_l$$
$$\mathrm{vec}\,(0 \oplus M) = \sum_{l=0}^{k-1} e_1 \otimes e_l \otimes e_1 \otimes c_l$$

where c_l is the l-th column of M. In our case, M is always some power of the k dimensional generalized not operator. With this in mind we can write

$$\mathrm{vec}\,(E_{ii} \otimes X^{a_i}) = \sum_{l=0}^{k-1} e_i \otimes e_l \otimes e_i \otimes e_{a_i+l}$$

where i can be 0 or 1 since it is used to effectively index the digits of a string and we are only dealing with strings of length two. This expression helps to express each term in (11) in a clear and informative way as

$$\mathrm{vec}\,(E_{ii} \otimes X^{a_i})\,\mathrm{vec}\,(E_{i'i'} \otimes X^{a_{i'}})^* = \sum_{l,l'=0}^{k-1} E_{ii'} \otimes E_{ll'} \otimes E_{ii'} \otimes E_{a_i+l,a'_i+l'}.$$

This means

$$J_x = \sum_{i,i'=0}^{n-1} \sum_{l,l'=0}^{k-1} E_{ii'} \otimes E_{ll'} \otimes E_{ii'} \otimes E_{a_i+l,a'_i+l'}.$$

While A_s are simply

$$\begin{aligned} A_s &= \sum_{x\in\zeta_s} \sum_{i,i'=0}^{n-1} \sum_{l,l'=0}^{k-1} E_{ii'} \otimes E_{ll'} \otimes E_{ii'} \otimes E_{a_i+l,a'_i+l'} \\ &= \sum_{i,i'=0}^{n-1} \sum_{l,l'=0}^{k-1} E_{ii'} \otimes E_{ll'} \otimes E_{ii'} \otimes f_{A_s}(i,i',l,l') \end{aligned} \tag{12}$$

where

$$\begin{aligned} f_{A_s}(i,i',l,l') &= \sum_{x\in\zeta_s} E_{a_i+l,a'_i+l'} \\ &= \begin{cases} X^{(l-l')} \text{ if } i = i' \\ FX^{s+(l-l')} \text{ if } i \neq i' \end{cases} \end{aligned}$$

and

$$F = \begin{pmatrix} 0 & & 1 \\ 0 & \diagup & 0 \\ 1 & & 0 \end{pmatrix}$$

is the generalized flip operator on k dimensions. Observe that the first three tensor product terms can be simply thought of as placement operators. which place each particular $f_{A_s}(i,i',l,l')$ in a specific block since they are all non-overlapping for different values of i,i',l,l'. Therefore A_s can be thought of as being composed of four different blocks — one corresponding to each possible combination of (i,i') from the set $\{0,1\}$ since $n=2$. Also observe, that we can expand the sum in (12) with respect to l and l' and recombine according to blocks to get

$$A_s = \begin{pmatrix} C_{A_s} & B_{A_s} \\ B^*_{A_s} & D_{A_s} \end{pmatrix}$$

where

$$C_{A_s} = \sum_{a=0}^{k-1} X^a \otimes E_{00} \otimes X^a$$
$$B_{A_s} = \sum_{a=0}^{k-1} FX^a \otimes E_{01} \otimes FX^{a+s}$$
$$D_{A_s} = \sum_{a=0}^{k-1} X^a \otimes E_{11} \otimes X^a.$$

Knowing this structure of A_s where only B and B^* (the blocks which correspond to $i \neq i'$) vary with respect to s, will enable us to propose the following, explicit construction of $k^2 Y_2$.

$$k^2 Y_2 = \begin{pmatrix} C_{k^2 Y_2} & B_{k^2 Y_2} \\ B^*_{k^2 Y_2} & D_{k^2 Y_2} \end{pmatrix}$$

where

$$C_{k^2 Y_2} = \sum_{a=0}^{k-1} X^a \otimes E_{00} \otimes \left(2X^a - \frac{1}{k} \mathcal{J}_k \right) \tag{13}$$
$$B_{k^2 Y_2} = \mathcal{J}_k \otimes E_{01} \otimes \frac{1}{k} \mathcal{J}_k$$
$$D_{k^2 Y_2} = \sum_{a=0}^{k-1} X^a \otimes E_{11} \otimes \left(2X^a - \frac{1}{k} \mathcal{J}_k \right).$$

Here, $\mathcal{J}_k$ is the all ones matrix, and X^a is the finite shift operator raised to the a-th power, both of dimension k.

To prove that the theorem holds, we must show that $Y_2 - \frac{1}{m} A_s \geq 0$, and that $|| \operatorname{Tr}_{\mathcal{Y}} Y_2 ||_\infty = \frac{2}{k}$.

SHOWING THAT $Y_2 - \frac{1}{m}A_s \geq 0$: Proving $Y_2 - \frac{1}{k^2}A_s \geq 0$ is equivalent to proving $k^2 Y_2 - A_s \geq 0$. With the form of A_s and $k^2 Y_2$ in mind, $k^2 Y_2 - A_s$ is then

$$k^2 Y_2 - A_s = \begin{pmatrix} C & B \\ B^* & D \end{pmatrix}$$

where

$$\begin{aligned}
C = C_{k^2 Y_2} - C_{A_s} &= \sum_{a=0}^{k-1} X^a \otimes E_{00} \otimes \left(X^a - \frac{1}{k}\mathcal{J}_k \right) \\
B = B_{k^2 Y_2} - B_{A_s} &= \sum_{a=0}^{k-1} F X^a \otimes E_{01} \otimes \left(-F X^{a+s} + \frac{1}{k}\mathcal{J}_k \right) \\
D = D_{k^2 Y_2} - D_{A_s} &= \sum_{a=0}^{k-1} X^a \otimes E_{11} \otimes \left(X^a - \frac{1}{k}\mathcal{J}_k \right).
\end{aligned}$$

Suppose we have some Hermitian operator Q and want to show that it is positive semidefinite. If we know Q is Hermitian, then if $Q^2 = \lambda Q$ for $\lambda \geq 0$, then we have $Q \geq 0$. This follows from the fact that $Q^2 = QQ = QQ^* = \lambda Q$ implies that $Q = \frac{1}{\sqrt{\lambda}} Q \frac{1}{\sqrt{\lambda}} Q^*$ which is equivalent to $Q \geq 0$. Since $k^2 Y_2 - A_s$ given blockwise above is clearly Hermitian, we proceed to show that $(k^2 Y_2 - A_s)^2 = \lambda (k^2 Y_2 - A_s)$ for $\lambda = 2k$.

First, observe the following identities.

$$\begin{aligned}
\left(X^a - \frac{1}{k}\mathcal{J}_k \right) \left(X^b - \frac{1}{k}\mathcal{J}_k \right) &= X^{a+b} - \frac{1}{k}\mathcal{J}_k \\
\left(-F X^a + \frac{1}{k}\mathcal{J}_k \right) \left(-F X^b + \frac{1}{k}\mathcal{J}_k \right) &= F X^a F X^b - \frac{1}{k}\mathcal{J}_k \\
&= X^{b-a} - \frac{1}{k}\mathcal{J}_k \\
\left(-F X^a + \frac{1}{k}\mathcal{J}_k \right) \left(X^b - \frac{1}{k}\mathcal{J}_k \right) &= X^a F X^b + \frac{1}{k}\mathcal{J}_k \\
&= F X^{b-a} - \frac{1}{k}\mathcal{J}_k
\end{aligned}$$

$$\left(X^a - \frac{1}{k}\mathcal{J}_k\right)\left(-FX^b + \frac{1}{k}\mathcal{J}_k\right) = FX^aX^b + \frac{1}{k}\mathcal{J}_k$$
$$= FX^{a+b} - \frac{1}{k}\mathcal{J}_k$$

They follow from the fact that $\mathcal{J}_k$ is invariant under multiplication by X^a or FX^a and that $\frac{1}{k^2}\mathcal{J}_k^2 = \frac{1}{k}\mathcal{J}_k$. Additionally, note that when the sum is taken over all a and b in $\mathbb{Z}_k$ of a term like X^{a+b}, we get

$$\sum_{a,b} X^{a+b} = k\sum_a X^a.$$

Now,

$$(k^2Y_2 - A_s)^2 = \begin{pmatrix} C & B \\ B^* & D \end{pmatrix}\begin{pmatrix} C & B \\ B^* & D \end{pmatrix} = \begin{pmatrix} CC + BB^* & CB + BD \\ B^*C + DB^* & B^*B + DD \end{pmatrix}$$

where using the identities above, we get

$$\begin{aligned} CC &= \left(\sum_{a=0}^{k-1} X^a \otimes E_{00} \otimes \left(X^a - \frac{1}{k}\mathcal{J}_k\right)\right)\left(\sum_{b=0}^{k-1} X^b \otimes E_{00} \otimes \left(X^b - \frac{1}{k}\mathcal{J}_k\right)\right) \\ &= \sum_{a=0,b=0}^{k-1} X^aX^b \otimes E_{00} \otimes \left(X^a - \frac{1}{k}\mathcal{J}_k\right)\left(X^b - \frac{1}{k}\mathcal{J}_k\right) \\ &= k\sum_{a=0}^{k-1} X^a \otimes E_{00} \otimes \left(X^a - \frac{1}{k}\mathcal{J}_k\right) \\ &= kC \end{aligned}$$

$$\begin{aligned} BB^* &= \left(\sum_{a=0}^{k-1} FX^a \otimes E_{01} \otimes \left(-FX^{a+s} + \frac{1}{k}\mathcal{J}_k\right)\right)\left(\sum_{b=0}^{k-1} FX^b \otimes E_{10} \otimes \left(-FX^{b+s} + \frac{1}{k}\mathcal{J}_k\right)\right) \\ &= \sum_{a=0,b=0}^{k-1} FX^aFX^b \otimes E_{00} \otimes \left(-FX^{a+s} + \frac{1}{k}\mathcal{J}_k\right)\left(-FX^{b+s} + \frac{1}{k}\mathcal{J}_k\right) \\ &= k\sum_{a=0}^{k-1} X^a \otimes E_{00} \otimes \left(X^a - \frac{1}{k}\mathcal{J}_k\right) \\ &= kC \end{aligned}$$

and similarly,

$$DD = kD$$
$$B^*B = kD.$$

Also, we have

$$\begin{aligned}
CB &= \left(\sum_{a=0}^{k-1} X^a \otimes E_{00} \otimes \left(X^a - \frac{1}{k}\mathcal{J}_k\right)\right)\left(\sum_{b=0}^{k-1} FX^b \otimes E_{01} \otimes \left(-FX^{b+s} + \frac{1}{k}\mathcal{J}_k\right)\right) \\
&= \sum_{a=0,b=0}^{k-1} X^a FX^b \otimes E_{01} \otimes \left(X^a - \frac{1}{k}\mathcal{J}_k\right)\left(-FX^{b+s} + \frac{1}{k}\mathcal{J}_k\right) \\
&= k\sum_{a=0}^{k-1} FX^a \otimes E_{01} \otimes \left(-FX^{a+s} + \frac{1}{k}\mathcal{J}_k\right) \\
&= kB
\end{aligned}$$

$$\begin{aligned}
BD &= \left(\sum_{a=0}^{k-1} FX^a \otimes E_{01} \otimes \left(-FX^{a+s} + \frac{1}{k}\mathcal{J}_k\right)\right)\left(\sum_{b=0}^{k-1} X^b \otimes E_{11} \otimes \left(X^b - \frac{1}{k}\mathcal{J}_k\right)\right) \\
&= \sum_{a=0,b=0}^{k-1} FX^a X^b \otimes E_{01} \otimes \left(-FX^{a+s} + \frac{1}{k}\mathcal{J}_k\right)\left(X^b - \frac{1}{k}\mathcal{J}_k\right) \\
&= k\sum_{a=0}^{k-1} FX^a \otimes E_{01} \otimes \left(-FX^{a+s} + \frac{1}{k}\mathcal{J}_k\right) \\
&= kB
\end{aligned}$$

and similarly,

$$B^*C = kB^*$$
$$DB^* = kB^*$$

so we get that

$$(k^2Y_2 - A_s)^2 = \begin{pmatrix} CC + BB^* & CB + BD \\ B^*C + DB^* & B^*B + DD \end{pmatrix} = 2k\begin{pmatrix} C & B \\ B^* & D \end{pmatrix} = 2k(k^2Y_2 - A_s).$$

It follows that $k^2Y_2 - A_s$, or $Y_2 - \frac{1}{k^2}A_s$ must be positive semidefinite.

Showing That $||\operatorname{Tr}_{\mathcal{Y}} Y_2||_\infty = \frac{2}{k}$ For All k: In (12) we had a very neat expression for what each A_s must be. Analysis on this expression lead to viewing A_s as being split into four blocks, and constructing Y_2 to match them. In this section, we will go from the four block representation of $k^2 Y_2$ to the a similar expression as the one we started with for A_s. From (12) and (13) we see that $k^2 Y_2$ can be written as

$$k^2 Y_2 = \sum_{i,i'=0}^{n-1} \sum_{l,l'=0}^{k-1} E_{ii'} \otimes E_{ll'} \otimes E_{ii'} \otimes f_{k^2 Y_2}(i, i', l, l') \tag{14}$$

where

$$f_{k^2 Y_2}(i, i', l, l') = \begin{cases} 2X^{(l-l')} - \frac{1}{k}\mathcal{J}_k & \text{if } i = i' \\ \frac{1}{k}\mathcal{J}_k & \text{if } i \neq i' \end{cases}.$$

We write $k^2 Y_2$ like this so that the partial trace computation becomes more straight forward.

$$\begin{aligned} \operatorname{Tr}_{\mathcal{Y}}(k^2 Y_2) &= \sum_{i,i'=0}^{n-1} \sum_{l,l'=0}^{k-1} \operatorname{Tr}_{\mathcal{Y}}(E_{ii'}) \otimes \operatorname{Tr}_{\mathcal{Y}}(E_{ll'}) \otimes E_{ii'} \otimes f_{k^2 Y_2}(i, i', l, l') \\ &= E_{00} \otimes k\left(2X^0 - \frac{1}{k}\mathcal{J}_k\right) + E_{11} \otimes k\left(2X^0 - \frac{1}{k}\mathcal{J}_k\right) \\ &= (E_{00} + E_{11}) \otimes k\left(2\mathbb{I} - \frac{1}{k}\mathcal{J}_k\right) \\ &= \mathbb{I}_n \otimes (2k\mathbb{I}_k - \mathcal{J}_k) \end{aligned}$$

From the single query dual we know that y_1, the dual objective value, must be bounded below according to the constraint

$$y_1 \mathbb{I}_{\mathcal{X}} = y_1 \mathbb{I}_n \otimes \mathbb{I}_k \geq \operatorname{Tr}_{\mathcal{Y}} Y_2.$$

From the computation of $\operatorname{Tr}_{\mathcal{Y}} k^2 Y_2$ above, it follows that y_1 must satisfy

$$k^2 y_1 \mathbb{I}_n \otimes \mathbb{I}_k \geq \mathbb{I}_n \otimes (2k\mathbb{I}_k - \mathcal{J}_k) \tag{15}$$

But,

$$\mathbb{I}_n \otimes \lambda \mathbb{I}_k \geq \mathbb{I}_n \otimes (2k\mathbb{I}_k - \mathcal{J}_k)$$

if and only if $\lambda \geq ||2k\mathbb{I}_k - \mathcal{J}_k||_\infty = 2k$. Therefore we get that (15) holds if and only if $y_1 \geq \frac{2k}{k^2}$. Choosing y_1 to be $\frac{2}{k}$ gives an objective value of the single query dual that matches the probability of success of the Meyer and Pommersheim algorithm when it makes a single query. This implies the algorithm is optimal when making single queries to hidden strings of length two with alphabets of arbitrary size and that the theorem above holds.

□

2.2.3 Insight From the Proof

This section contains some more high level discussion on what the proof from the preceding section implies. We now have a tight bound on the probability of success for single query quantum algorithms aiming to solve the general sum problem. This by no means closes the book on the general sum problem as a whole however, since multiple adaptive quantum queries can yield a much better probability of success, as the Meyer and Pommersheim algorithm illustrates. Ideally we would like to say something about the optimal probability of success in cases where multiple quantum queries are made adaptively. The proof in the preceding section sheds crucial light on the structure of the sum classes, A_s. Exploitation of this structure is what led to constructing Y_2 which in turn gave a dual objective value matching the existing algorithm for single queries. If we take a moment to discuss this structure and see how it changes as the number of queries increases, perhaps we can again constructively come up with dual variables that yield insightful objective values.

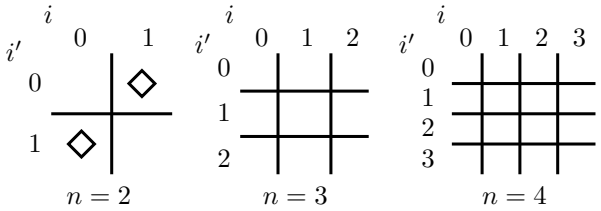

Figure 2.3: Variance and placement changes for different values of n, arbitrary k, and one query. A rhombus indicates variance with respect to s. A blank square indicates no variance.

Each A_s is given by equation (12). The first three tensor product terms simply place the forth term in a specific location based on the values of i, i', l, and l'. The l and l' range over the alphabet, while the i and i' range over the string indices. Hence the first three terms are referred to as *placement operators* while the fourth term is referred to

as the *string operator*, because it is dependent of the string a particular J_x represents. This equation arose from the definition of each Choi representation for all possible string oracles given an alphabet size and a string length. If the alphabet size or the string length increases, the first three terms in (12) will simply place the last terms into a more varied partition structure. For instance, if we look at the first term, it splits each A_s into the four blocks we analyzed. Each of these blocks denotes a different combination of indicies. Since we were working with only $n = 2$ there were four total. If n increases to 3 there will be nine blocks in each A_s. In the $n = 2$ case, we noticed that the last tensor product term exhibited variance with respect to s when $i \neq i'$. The term *variance* is used here to suggest that there is some average operator that is equidistant to each of the operators $E_{a_i+l,a'_i+l'}$ which exhibit symmetric variation with respect to s. This is illustrated in figure 2.3. If $n = 3$ none the blocks will have *any* variance in the last tensor product term. This is because if one enumerates all possible strings of size 3 and some arbitrary finite alphabet, and looks at subsets containing only two of the three digits for strings in a particular sum class, they will get the same set regardless of which sum class was chosen. This is the heart of the general sum problem multi query dual. If the queries get increased to 2, then, in a sense, there will be variance between the tensor product factors of each A_s when $n = 3$. The farther apart all the A_ss are the higher the minimum value of y_1 will be raised by the dual variables in order for them to be feasible. To examine this behavior in a more precise manner, we must first explore how the dual changes when more than a single query is made. This is done in the next chapter.

Chapter 3

Multi Query Algorithms

This chapter provides speculation on the general case where multiple adaptive queries can be made.

3.1 Extending the SDP to multiple Queries

In this section the SDP from 2.1.3 is extended to work for multiple queries. This SDP already deals with any arbitrary values of n and k — which only change the dimension of the Choi matrix constants in the problem. It turns out that making additional queries will keep the same general form of the dual and primal, with some additional constraints. The Choi matrix constants that the P_a's in the primal and Y_2 in the dual are constrained against will also change; in effect increasing the dimension of those variables even further based on the number of queries made. For the sake of derivation, the SDP for two queries and arbitrary n and k will be our aim initially. Generalization from this SDP to SDPs dealing with arbitrarily larger number of queries will be straightforward and discussed as we go along.

The diagram in figure 3.1 illustrates what is happening when two queries are allowed. Initially the algorithm prepares and sends some start state to the oracle. Then, the oracle returns some state. The algorithm can now perform any valid quantum operation, including measurement, or entangling operations with its work space qubits. Then a new quantum state is sent to the oracle. The oracle again returns some state by acting as it is defined in section 1.3. Now the algorithm can perform any post-processing, and must perform a measurement to collapse the result onto some classically identifiable state that tells us

what the sum of the hidden string is. Note that the oracle operator is unchanged from one instance to the next. We have to reflect this intermediate channel the algorithm can apply to the result of the first query in our SDP. We show how this change is reflected in the primal and derive the changes in the dual from there.

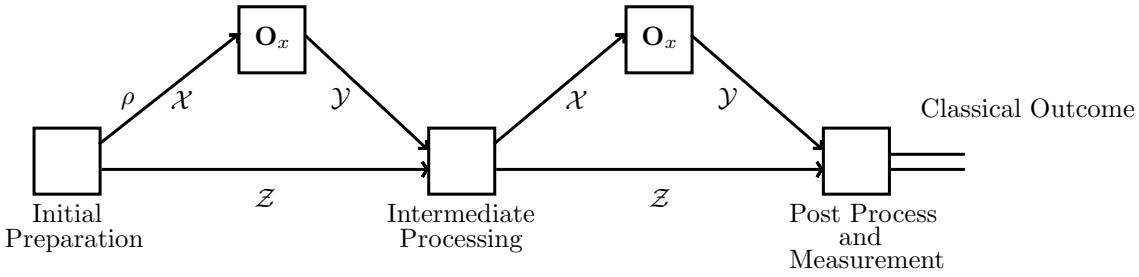

Figure 3.1: The general picture of any quantum algorithm making two calls to the oracle.

3.1.1 Two Query SDP

We are now comparing a larger and more complicated interactive measurement to two oracle calls. This leads to two changes to the primal shown in section 2.1.3. First, our SDP primal will now be constrained by the overlap of each measurement operator with *two* copies of each Choi representation: $J_x \otimes J_x = J_x^{\otimes 2}$. The second consequence reflects making sure these measurement operators adhere to the proper and more complicated structure imposed by figure 3.1 for general two-query algorithms.

We have four equivalent spaces now, which are named differently so we can distinguish between them. Let the first oracle live in $\mathrm{L}(\mathcal{Y}_1 \otimes \mathcal{X}_1)$ and the second live in $\mathrm{L}(\mathcal{Y}_2 \otimes \mathcal{X}_2)$. The sum of all operators $P_0, \ldots, P_{k-1}$ must be equal to the identity on $\mathcal{Y}_2$ tensored with some positive semidefinite operator R such that $\mathrm{Tr}_{\mathcal{X}_2}(R) = \mathbb{I}_{\mathcal{Y}_1} \otimes \rho$ where $\rho \in \mathrm{D}(\mathcal{X}_1)$ is the initial state. Therefore there are now two constraints that dictate the structure of the P_as.

$$\sum_{i=1}^{k} P_i = \mathbb{I}_{\mathcal{Y}_2} \otimes R$$
$$\mathrm{Tr}_{\mathcal{X}_2}(R) = \mathbb{I}_{\mathcal{Y}_1} \otimes \rho$$

where $P_i \in \mathrm{L}(\mathcal{Y}_2 \otimes \mathcal{X}_2 \otimes \mathcal{Y}_1 \otimes \mathcal{X}_1)$ and $R \in \mathrm{L}(\mathcal{X}_2 \otimes \mathcal{Y}_1 \otimes \mathcal{X}_1)$. Here R can be thought of what freedom the quantum algorithm has at the intermediate step in figure 3.1. Really,

the only freedom that the quantum algorithm has at that stage lies in the subsystem $\mathcal{X}_2$. Therefore if $\mathcal{X}_2$ is traced out from R we must get $\mathbb{I}_{\mathcal{Y}_1} \otimes \rho$ as in the one query case.

The primal for two queries is

$$\begin{aligned} &\text{maximize} && \gamma \\ &\text{subject to} && \gamma \leq \left\langle P_{\mathrm{SUM}(x)}, J_x^{\otimes 2} \right\rangle \text{ for all } x \in \mathbb{Z}_k^n \\ & && \sum_{i=1}^{k} P_i = \mathbb{I}_{\mathcal{Y}_2} \otimes R \\ & && \mathrm{Tr}_{\mathcal{X}_2}(R) = \mathbb{I}_{\mathcal{Y}_1} \otimes \rho \\ & && R, \{P_i\} \in \mathrm{Pos}(\mathcal{X}). \end{aligned} \tag{16}$$

The corresponding dual is

$$\begin{aligned} &\text{minimize} && y_1 \\ &\text{subject to} && \sum_{i \in [m]} w_i \geq 1 \\ & && Y_3 \geq \left(\sum_{x \in \zeta_s} w_{i(x)} J_x^{\otimes 2} \right) \ \forall s \in \mathbb{Z}_k \\ & && \mathbb{I}_{\mathcal{X}_2} \otimes Y_2 \geq \mathrm{Tr}_{\mathcal{Y}_2}(Y_3) \\ & && y_1 \mathbb{I}_{\mathcal{X}_1} \geq \mathrm{Tr}_{\mathcal{Y}_1}(Y_2) \end{aligned} \tag{17}$$

where

$$\begin{aligned} &y_1 \in \mathbb{R} \\ &Y_2 \in \mathrm{Herm}(\mathcal{Y}_1 \otimes \mathcal{X}_1) \\ &Y_3 \in \mathrm{Herm}(\mathcal{Y}_2 \otimes \mathcal{X}_2 \otimes \mathcal{Y}_1 \otimes \mathcal{X}_1) \\ &w_i \geq 0. \end{aligned}$$

This dual was derived using precisely the same technique shown for the one query case, so it is not repeated here. Note that the main difference between the one and two query dual programs is the tensor power of 2 on each J_x, and the extra inequality constraint. This turns into a pattern as for q queries, each J_x is raised to the q-th tensor power, and there are q inequalities following the pattern shown in the two query case. These properties are reflected in the SDP for arbitrarily large q.

3.1.2 Multi Query Primal

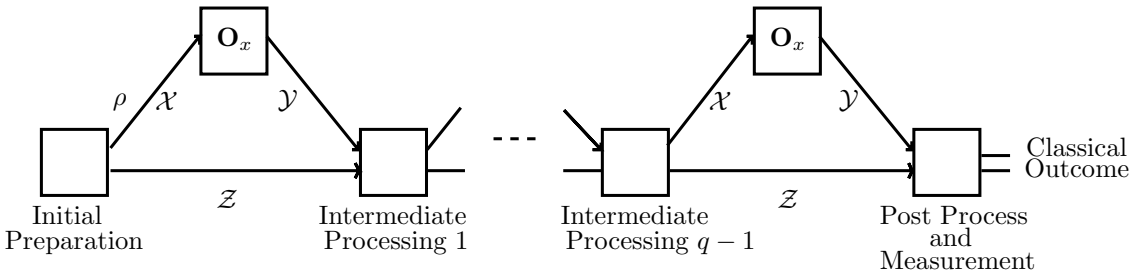

Figure 3.2: The general picture of any quantum algorithm making q calls to the oracle.

In this section general SDP form for q queries is given. This SDP extends very naturally from the two query case. The diagram in figure 3.2 depicts what any arbitrary q query quantum algorithm looks like. This picture is a straightforward extension of the one in figure 3.1. The initial state is still some valid density operator which the algorithm has prepared, while each intermediate stage will introduce another positive semidefinite variable like the variable R for the two query case. We will have a series of R variables $\{R_i\}$, where i will ranged from 1 to $q-1$. The interactive measurement collection now lives in a very large space as it must overlap $J_x^{\otimes q}$. There are still k operators that compose this interactive measurement corresponding to each of the k outcomes of the SUM function on a string in $\mathbb{Z}_k^n$. The Choi representation of each string x lives in the space $\mathrm{L}(\mathcal{Y} \otimes \mathcal{X})$ as discussed in section REF. Therefore the interactive measurement operators $\{P_i\}$ live in the space $\mathrm{Pos}((\mathcal{Y} \otimes \mathcal{X})^{\otimes q}$. Recall that $\mathcal{X} = \mathcal{Y}$ but are called differently to distinguish the input space of some string x's oracle from its output space. Along similar lines, if we want to keep the spaces of each call to the oracle distinct, we introduce subscripts so that $\mathrm{Pos}((\mathcal{Y} \otimes \mathcal{X})^{\otimes q}$ is actually $\mathrm{Pos}(\mathcal{Y}_q \otimes \mathcal{X}_q \otimes \cdots \otimes \mathcal{Y}_1 \otimes \mathcal{X}_1)$. In terms of thinking about their dimension, these two notations are absolutely equivalent. Since $\dim \mathcal{Y}_i = \dim \mathcal{X}_i = nk$, $\dim \mathcal{P}_i \in \mathrm{Pos}((\mathcal{Y} \otimes \mathcal{X})^{\otimes q} = (nk)^{2q}$. It is clear that the dimensions of the interactive measurement operators increase quite drastically when even a single query is made above one, making the solution to SDPs for such cases intractable.

When constructing our primal, as usual, we first turn to thinking about how the constraints have changed. The group of constraints that enforce the SUM function remain the same, they provide an upper bound for γ, the probability of success, in terms of the overlap between each string's Choi representation tensored q times, with the interactive

measurment operator that corresponds to the strings sum.

$$\begin{aligned}\gamma &\leq \left\langle J_{x_1}^{\otimes q}, P_{\mathrm{SUM}(x_1)} \right\rangle \\ &\vdots \\ \gamma &\leq \left\langle J_{x_m}^{\otimes q}, P_{\mathrm{SUM}(x_m)} \right\rangle\end{aligned}$$

The second group of constraints impose the desired structure on our collection of positive semidefinite operators that comprise the interactive measurement. This advanced restriction on their forms ensures that the operations performed at each intermediate position in figure 3.2 are valid within the quantum setting and comes from [8] and [7]. These constraints form an inequality ladder which looks as follows.

$$\begin{aligned}\sum_{i=1}^{k} P_i &= \mathbb{I}_{\mathcal{Y}_q} \otimes R_{q-1} \\ \mathrm{Tr}_{\mathcal{X}_q}(R_{q-1}) &= \mathbb{I}_{\mathcal{Y}_{q-1}} \otimes R_{q-2} \\ \mathrm{Tr}_{\mathcal{X}_{q-1}}(R_{q-2}) &= \mathbb{I}_{\mathcal{Y}_{q-2}} \otimes R_{q-3} \\ &\vdots \\ \mathrm{Tr}_{\mathcal{X}_2}(R_2) &= \mathbb{I}_{\mathcal{Y}_2} \otimes R_1 \\ \mathrm{Tr}_{\mathcal{X}_1}(R_1) &= \mathbb{I}_{\mathcal{Y}_1} \otimes \rho\end{aligned}$$

where $R_i \in \mathrm{Pos}(\mathcal{X}_{i+1} \otimes (\mathcal{Y}_i \otimes \mathcal{X}_i) \otimes \cdots \otimes (\mathcal{Y}_1 \otimes \mathcal{X}_1))$, $\{\} \in \mathrm{Pos}(\mathcal{Y}_q \otimes \mathcal{X}_q) \otimes \cdots \otimes (\mathcal{Y}_1 \otimes \mathcal{X}_1))$, and $\rho \in \mathrm{D}(\mathcal{X})$.

Putting these two groups together, we get that the primal problem which solves for the

optimal probability of success for n, k, and $q \leq n$ queries.

$$\begin{aligned}
&\text{maximize} && \gamma \\
&\text{subject to} && \gamma \leq \left\langle P_{\mathrm{SUM}(x)}, J_x^{\otimes q} \right\rangle \text{ for all } x \in \mathbb{Z}_k^n \\
& && \sum_{i=1}^{k} P_i = \mathbb{I}_{\mathcal{Y}_q} \otimes R_{q-1} \\
& && \mathrm{Tr}_{\mathcal{X}_q}(R_{q-1}) = \mathbb{I}_{\mathcal{Y}_{q-1}} \otimes R_{q-2} \\
& && \mathrm{Tr}_{\mathcal{X}_{q-1}}(R_{q-2}) = \mathbb{I}_{\mathcal{Y}_{q-2}} \otimes R_{q-3} \\
& && \vdots \\
& && \mathrm{Tr}_{\mathcal{X}_2}(R_2) = \mathbb{I}_{\mathcal{Y}_2} \otimes R_1 \\
& && \mathrm{Tr}_{\mathcal{X}_1}(R_1) = \mathbb{I}_{\mathcal{Y}_1} \otimes \rho \\
& && \{R_{q-1}, \ldots, R_1\}, \{P_i\}, \rho \geq 0
\end{aligned} \tag{18}$$

The corresponding dual is

$$\begin{aligned}
&\text{minimize} && y_1 \\
&\text{subject to} && \sum_{i \in [m]} w_i \geq 1 \\
& && Y_{q+1} \geq \left(\sum_{x \in \zeta_0} w_{i(x)} J_x^{\otimes q} \right) \quad \forall s \in \mathbb{Z}_k \\
& && \mathbb{I}_{\mathcal{X}_q} \otimes Y_q \geq \mathrm{Tr}_{\mathcal{Y}_q}(Y_{q+1}) \\
& && \mathbb{I}_{\mathcal{X}_{q-1}} \otimes Y_{q-1} \geq \mathrm{Tr}_{\mathcal{Y}_{q-1}}(Y_q) \\
& && \vdots \\
& && \mathbb{I}_{\mathcal{X}_2} \otimes Y_2 \geq \mathrm{Tr}_{\mathcal{Y}_2}(Y_3) \\
& && y_1 \mathbb{I}_{\mathcal{X}_1} \geq \mathrm{Tr}_{\mathcal{Y}_1}(Y_2)
\end{aligned}$$

where

$$\begin{aligned}
&y_1 \in \mathbb{R} \\
&Y_i = Y_i^* \\
&w_i \geq 0.
\end{aligned}$$

This dual can be viewed as having three separate sections. First, there are some weights w_i which are positive real numbers and are each paired up with a unique string in $\mathbb{Z}_k^n$. Second, there is a big operator Y_{q+1}. Y_{q+1} needs to be greater than or equal to the sum over the weighted strings of each sum class. Recall that a sum class was defined as all strings for which the sum function has an equivalent value. In a sense, these constraints place a lower bound on the spectral norm of Y_{q+1}. The third and final section of constraints is an inequality ladder. This ladder cuts down a system $\mathcal{Y}_i$ from a big operator in $\mathrm{Pos}((\mathcal{Y}\otimes\mathcal{X})^{\otimes i})$ and bounds a smaller operator in $\mathrm{Pos}((\mathcal{Y}\otimes\mathcal{X})^{\otimes i-1})$ by first tensoring the identity on $\mathcal{X}_i$ to it. This goes from $Y_{q+1}\in\mathrm{Pos}((\mathcal{Y}\otimes\mathcal{X})^q)$ all the way down to $y_1\in\mathbb{R}$. The complex Euclidean vector spaces $\mathcal{Y}_i$ and $\mathcal{X}_i$ are all equivalent to one another, so order doesn't really matter as long as consistency is maintained throughout to create the unique stages of processing of the algorithm outlined in figure 3.2.

3.2 Multi Query Results

Numerically there is little hope of analysis for this SDP. Many reduction measures were conceived and taken but none proved to be tractable for even the smallest of nontrivial cases. The dimensions of the matrices at hand explode when even two queries are made. If we let $n=4$, $k=4$, and $q=2$ then the dimension of Y_{q+1} becomes 16^4 which was intractable on a machine with 128 gigabytes of memory. There are several reductions one can make. First, we can enforce a uniform prior distribution over the possible strings. This eliminates the variable W as it becomes simply $\frac{1}{m}\mathbb{I}_{\mathbb{C}^m}$. This allows us to pull the w_i weights out in front of each sum to get the operators denoted A_s in the one query case: $\frac{1}{m}\sum_{x\in\zeta_s} J_x^{\otimes q}$.

Second, we can enforce $\mathrm{Tr}_{\mathcal{Y}_q}(Y_q)$ to have the particular structure $c\cdot\mathbb{I}$, where c is some constant. This collapses the inequality ladder in the multi-query dual to the single constraint

$$y_1 \geq \frac{||\,\mathrm{Tr}_{\mathcal{Y}_q}(Y_{q+1})||_\infty}{(q-1)\dim(\mathcal{Y})}.$$

Justification for this assumption stems from the fact that the partial trace of Y_{q+1} when $q=1$ as constructed in section ref is always of the form $c\cdot\mathbb{I}$. This eliminates much of the structure of the problem however and isn't a favored reduction.

The need to reduce this problem further gave rise to two other ideas for approaches.

3.2.1 The Pretty Good Measurment

If we forget for a moment about the inequality ladder and focus on finding Y_{q+1} the problem becomes as follows: Minimize the spectral norm of $\mathrm{Tr}_{\mathcal{Y}_q} Y_{q+1}$ while satisfying $Y_{q+1} \geq \frac{1}{m} A_s$ for all $s \in \mathbb{Z}_k$. Since Y_{q+1} and its partial trace over $\mathcal{Y}_q$ is positive semidefinite its trace must be an upper bound for the spectral norm of this partial trace. This comes from the fact eigen spectrum of the tensor product between two operators is comprised of the cartesian product of their respective eigenvalues. This gives the following SDP.

$$\begin{aligned} \text{minimize} \quad & \mathrm{Tr}(Y_{q+1}) \\ \text{subject to} \quad & Y_{q+1} \geq \frac{1}{m} A_0 \\ & \vdots \\ & Y_{q+1} \geq \frac{1}{m} A_{k-1} \\ & Y_{q+1} \geq 0. \end{aligned}$$

This SDP actually represents the SDP for finding the optimal measurement for some collection of operators. To explore our original problem, it may be worth finding the optimal measurement strategy for the collection A_s and plugging that into the inequality ladder to get bound for y_1. If we consider this the dual for the optimal measurement problem, then the primal is

$$\begin{aligned} \text{maximize} \quad & \left\langle \begin{pmatrix} X_0 & & \\ & \ddots & \\ & & X_{k-1} \end{pmatrix}, \frac{1}{m} \begin{pmatrix} A_0 & & \\ & \ddots & \\ & & A_{k-1} \end{pmatrix} \right\rangle \\ \text{subject to} \quad & \sum_{i=0}^{k-1} X_i = \mathbb{I} \\ & \{X_i\} \geq 0. \end{aligned}$$

This SDP was also not tractable for any nontrivial cases. However, in [10] the notion of a pretty good measurement is presented. A pretty good measurement has the form $\frac{1}{m} \sigma^{-\frac{1}{2}} A_s \sigma^{-\frac{1}{2}}$ for each X_i variable in the primal, where $\sigma = \frac{1}{km} \sum_{s \in \mathbb{Z}_k} A_s = \frac{1}{km} \sum_{x \in \mathbb{Z}_k^n} J_x^{\otimes q}$. Not all $J_x^{\otimes q}$ commute so the inverse must be performed on the entire sum when computing $\sigma^{-\frac{1}{2}}$. Furthermore, the Moore-Penrose pseudo inverse must be used because $\sum_{x \in \mathbb{Z}_k^n} J_x^{\otimes q}$ may be singular.

To proceed with evaluation of this approach one must ensure that each $\sigma^{-\frac{1}{2}} A_s \sigma^{-\frac{1}{2}}$ is positive semidefinite. Moreover, the pretty good measurement would only be useful if it is optimal for the A_s operators. This can be checked using the Holevo criterion for optimal measurements, which states:

> A finite collection X_i is an optimal measurement for some collection A_i if and only if:
>
> 1. $\sum_i A_i X_i$ is Hermitian
> 2. $\sum_i A_i X_i \geq A_i$ for all i.

If all of these requirements hold, it would be interesting to take the dual variable associated with this optimal solution, and plug it into the dual for the multi query SDP. A bound on y_1 can obtained which may provide insight into the problem.

3.2.2 Separable Measurements

As soon as we make more than one query, an arbitrary algorithm solving the general sum problem adheres to the behavior outlined in figure 3.2. What that algorithm does in between oracle calls is entirely up to its strategy. One possibility is to post-process and measure after each query then use that classical outcome to adapt the next query. Another, is to adapt future queries strictly through quantum manipulation. Yet another is to not have any adaptive behavior between queries which would be like making them all in parallel.

Let us think about the case when only two queries are made. Each J_x is positive semidefinite. In fact, every J_x is a rank one projector and can be written in the form $J_x = \lambda_x u_x u_x^*$ where $\lambda = nk$ and $u \in \mathcal{S}(\mathcal{Y} \otimes \mathcal{X})$. Therefore the operator A_s is separable between $\mathcal{Y}_1 \otimes \mathcal{X}_1$ and $\mathcal{Y}_2 \otimes \mathcal{X}_2$. Since this is the case, we may ask what happens if we restrict Y_3 to be in $\mathrm{Sep}(\mathcal{Y}_2 \otimes \mathcal{X}_2 : \mathcal{Y}_1 \otimes \mathcal{X}_1)$ or in the general case $\mathcal{Y}_{q+1}$ to be in $\mathrm{Sep}(\mathcal{Y}_q \otimes \mathcal{X}_q : \mathcal{Y}_1 \otimes \mathcal{X}_1)$. Because of its construction, each A_s will be symmetric over the systems it is separable. This means that A_s will have an identical operator in each separable system $\mathcal{X} \otimes \mathcal{Y}$. If an expression of this operator is found, then the problem can be reduced to finding the component of Y_{q+1} in a single system $\mathcal{Y} \otimes \mathcal{X}$ then finding the bound that places on y_1. This would drastically reduce the computational demand of the problem.

The downside to this approach is that Y_q is restricted to be separable across each $\mathcal{Y} \otimes \mathcal{X}$.

If this separability is enforced in the primal problem it would imply that the algorithm must not have any adaptive behavior from one query to the next.

3.2.3 Increasing Queries

In section 2.2.3 we examined how the structure of A_s changes based on n. The structure of A_s is what governs how low we can bound the spectral norm of $\mathrm{Tr}_{\mathcal{Y}_q}(Y_{q+1})$. We saw that as n increased but the queries remained fixed at 1, the invariance between each A_s disappeared. When each A_s is very similar to the rest, the trace norm that bounds y_1 below through the ladder of inequalities in the multi-query dual will be very low. On the other hand, if each A_s is a substantial distance apart from the rest, then an optimal Y_{q+1} will bound y_1 to a higher minimum value. When all A_s were equivalent, we had no choice but to let Y_{q+1} be equal to A_s. Then the spectral norm of $\mathrm{Tr}_{\mathcal{Y}}(Y_2)$ for an optimal Y_2 for one query is lowest. This behavior continues when multiple queries are made. In fact, increasing the number of queries creates more variance between the A_s operators. In this section, we examine exactly what determines variance between sum class operators A_s.

To begin with, we derive an explicit equation for each A_s based on the multi-query dual. In that dual, each A_s is constructed the same way as for a single query, except that each Choi representation of a string in the sum class taken to the q-th tensor power. Suppose $n = 2$, then we have the following equation for A_s.

$$A_s = \sum_{x \in \zeta_s} \sum_{i,i',j,j'=0}^{n-1} \sum_{l,l',m,m'=0}^{k-1} E_{ii'} \otimes E_{ll'} \otimes E_{ii'} \otimes E_{a_i+l,a'_i+l'} \otimes E_{jj'} \otimes E_{mm'} \otimes E_{jj'} \otimes E_{a_j+m,a'_j+m'}$$

Note that there are still two indices at each placement operator. The difference is that now, there is an extra set of placement and string operators. The two string operators, called this way since they are dependent on the value of strings in the sum class, are $E_{a_i+l,a'_i+l'}$ and $E_{a_j+m,a'_j+m'}$. Without loss of generality, we can set l, l', m, m' to 0 and examine $E_{a_i,a'_i} \otimes E_{a_j,a'_j}$. This is exactly what gives rise to variance between the A_s operators. If $n = 4$ and we only have one string operator, $\sum_{x \in \zeta_s} E_{a_i,a'_i}$ will be invariant with respect to a choice of s for any assignment of $i, i' \in \mathbb{Z}_n$. On the other hand, two string operators yield $\sum_{x \in \zeta_s} E_{a_i,a'_i} \otimes E_{a_j,a'_j}$, which will vary with respect to s. The variance here however, will be in the correlation between E_{a_i,a'_i} and E_{a_j,a'_j} when the sum over all strings in a sum class is taken. Figure 3.3 shows which assignments of i, i', j and j' induce variance when two queries are made for some different values of n and arbitrary k.

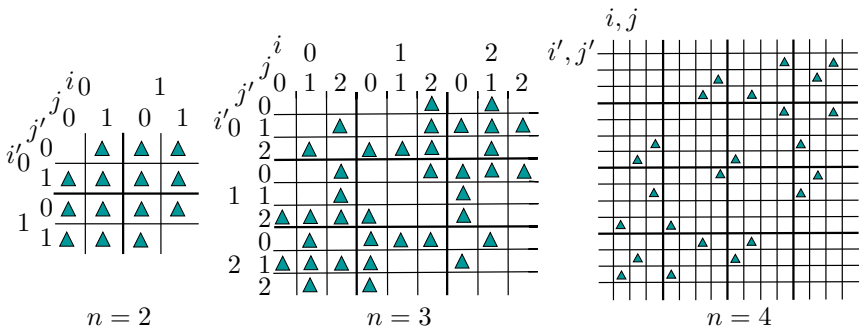

Figure 3.3: Variance tables for each A_s when two queries are made while the string length, n, is 2, 3, and 4. A triangle indicates variance with respect to s while a blank square indicates invariance.

What is left now, is to construct a feasible Y_{q+1} for any n, k and q, and plug that into the inequality ladder of the multi-query dual. The goal being to construct an operator Y_{q+1} that is equidistant from all the A_s and satisfies $k^n Y_{q+1} - A_s \geq 0$. This construction will involve strategic placements of the scaled all ones matrix and will give an upper bound on the probability of success for the general case. If this upper bound does not match the algorithm in [13], it suggests two possibilities. The first is that the algorithm is not optimal. The second is that Y_{q+1} is not optimal. To remove any doubt, a proof of optimality of Y_{q+1} or a primal objective value matching the dual value of Y_{q+1} would have to be supplied.

Chapter 4

Conclusion

In this thesis, the problem of success bounds for computing the sum function of a finite length string with arbitrary finite alphabet was treated. In Chapter 1 the problem was presented. In Chapter 2 a semidefinite program modeling the problem was derived and an upper bound for the probability of success of single query quantum algorithms was shown and proven. This bound happened to match the performance of an existing algorithm for the general sum problem. In Chapter 3 the road was paved towards computing closed form expressions for upper bounds on the probability of success of quantum algorithms making more than one sequential, adaptive queries. We hope that the work will be extended and a bound will be given for the two query case as well.

It was the author's choice to pursue the multi-query case rather than computing more probability of success upper bounds for different functions in the setting of non-adaptive quantum query algorithms. This problem would be easy to tackle as the content of chapter two could be applied in the same way, where the only difference would be that each equivalence class would no longer be a sum class but would have a different structure based on the function being considered.

The multi-query case is interesting because if the strategies discussed in Chapter 3 do lead to fruitful bounds for the sum function, this work could be used as a template to solve other functions on strings which are non-binary.

Going forward, there are several interesting steps to take by anyone interested in this problem. One is to continue hunting for a closed form of the dual optimum value for the two and then the multi-query sdp. Another interesting step which may provide further insight would be to apply the powerful theory of span programs to this problem. Using span programs, one can fix the probability of success and optimize instead over the minimum

number of queries needed to achieve it. This method can be used to say more about the optimality of the Meyer and Pommersheim algorithm.

References

[1] Scott Aaronson, Andris Ambainis, Kaspars Balodis, and Mohammad Bavarian. Weak parity. In *Automata, Languages, and Programming*, pages 26–38. Springer, 2014.

[2] Andris Ambainis, Kazuo Iwama, Akinori Kawachi, Hiroyuki Masuda, Raymond H Putra, and Shigeru Yamashita. Quantum identification of boolean oracles. In *STACS 2004*, pages 105–116. Springer, 2004.

[3] Gilles Brassard, Peter Hoyer, Michele Mosca, and booktitle=Quantum Computation and Quantum Information pages=53–74 year=2002 publisher=American Mathematical Society Tapp, Alain. Quantum amplitude amplification and estimation.

[4] Orest Bucicovschi, Daniel Copeland, David A Meyer, and James Pommersheim. Distinguishing symmetric quantum oracles and quantum group multiplication. *arXiv preprint arXiv:1503.05548*, 2015.

[5] Richard Cleve, Artur Ekert, Chiara Macchiavello, and Michele Mosca. Quantum algorithms revisited. *Proceedings of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences*, 454(1969):339–354, 1998.

[6] Edward Farhi, Jeffrey Goldstone, Sam Gutmann, and Michael Sipser. Limit on the speed of quantum computation in determining parity. *Physical Review Letters*, 81(24):5442, 1998.

[7] Gus Gutoski. *Quantum Strategies and Local Operations*. PhD thesis, University of Waterloo, 2009.

[8] Gus Gutoski and John Watrous. Toward a general theory of quantum games. In *Proceedings of the thirty-ninth annual ACM symposium on Theory of computing*, pages 565–574. ACM, 2007.

[9] Aram W Harrow and David J Rosenbaum. Uselessness for an oracle model with internal randomness. *Quantum Information & Computation*, 14(7&8):608–624, 2014.

[10] Paul Hausladen and William K Wootters. A pretty goodmeasurement for distinguishing quantum states. *Journal of Modern Optics*, 41(12):2385–2390, 1994.

[11] Phillip Kaye, Raymond Laflamme, and Michele Mosca. *An introduction to quantum computing*. Oxford University Press, Oxford UK, 1 edition, 2007.

[12] David A Meyer and James Pommersheim. Multi-query quantum sums. In *Theory of Quantum Computation, Communication, and Cryptography*, pages 153–163. Springer, 2014.

[13] David A Meyer and James Pommersheim. Multi-query quantum sums. In *Theory of Quantum Computation, Communication, and Cryptography*, pages 153–163. Springer, 2014.

[14] Abel Molina and John Watrous. Hedging bets with correlated quantum strategies. In *Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, page rspa20110621. The Royal Society, 2012.

[15] Peter W Shor. Algorithms for quantum computation: discrete logarithms and factoring. In *Foundations of Computer Science, 1994 Proceedings., 35th Annual Symposium on*, pages 124–134. IEEE, 1994.

[16] Wim van Dam. Quantum oracle interrogation: Getting all information for almost half the price. Proceedings of the 39th Annual IEEE Symposium on Foundations of Computer Science (FOCS), pages 362-367 (1998), 1998.

[17] John Watrous. Theory of quantum information. Unpublished Manuscript, 2014.