APPLICATION OF SDP TO PRODUCT RULES AND QUANTUM QUERY COMPLEXITY

by

RAJAT MITTAL

A dissertation submitted to the

Graduate School—New Brunswick

Rutgers, The State University of New Jersey

In partial fulfillment of the requirements

For the degree of

Doctor of Philosophy

Graduate Program in Computer Science

Written under the direction of

Mario Szegedy

And approved by

_____

_____

_____

_____

New Brunswick, New Jersey

May, 2011

ABSTRACT OF THE DISSERTATION

APPLICATION OF SDP TO PRODUCT RULES AND QUANTUM QUERY COMPLEXITY

By RAJAT MITTAL

Dissertation Director:

Mario Szegedy

In recent years, semidefinite programming has played a vital role in shaping complexity theory and quantum computing. There have been numerous applications ranging from estimating quantum values, over approximating combinatorial quantities, to proving various bounds. This work extends the use of semidefinite programs (SDPs) to proving product rules and to characterizing quantum query complexity.

In the first application, we provide a general framework to establishing product rules for quantities that can be expressed (or approximated) using SDPs. We use duality theory to give product rules, which bound the value of the "product" of two problems in terms of their value. Some previous results have implicitly used the properties of SDPs to give such product rules. Here we give sufficient and necessary conditions under which these approaches work, thereby enabling us to capture these previous results under our unified framework. We also include a discussion about alternate definitions of what a "product" means and how they fit into our approach.

The second application provides an SDP characterization of quantum query complexity, which is one of the ways in which complexity of a function can be measured. It is known that quantum query complexity can be lower bounded by the so-called "adversary method" which is expressible as a semidefinite program. Recently, Ben Reichardt showed that the adversary method leads to a tight lower bound for boolean functions

by converting the solution of this SDP (of adversary method) into an algorithm. We show that a related SDP, called "witness size" in this thesis, provides a tight bound on the quantum query complexity of non boolean functions (total as well as partial). This witness size SDP is also used to give composition results for quantum query complexity. We also show that the witness size is bounded by a constant multiple of the adversary bound.

Finally, we briefly explore whether other convex programming paradigms can be useful in complexity theory. One of them is copositive programming. We show that one of the recent result about parallel repetition of unique games, by Barak et.al., can be interpreted as an application of copositive programming.

Acknowledgement

First and most of all, I would like to thank my advisor Mario Szegedy for his enormous help and guidance throughout the PhD. He has been a constant source of inspiration and motivation. He made me realize the difference between knowing something and understanding it. The discussions with him were always fun whether academic or non-academic.

I also thank other theory faculty in Rutgers for their support, help, insights and intuition. The reading seminar and discussions with Mike Saks made me interested in areas such as algebraic complexity, algorithmic number theory and communication complexity. As a graduate advisor in the beginning and later as the part of my committee, William Steiger has guided me in many aspects of my PhD and I will always be thankful to him.

It was a privilege to work with my coauthors Troy Lee, Ben Reichardt and Robert Spalek. Troy has been a great mentor, coauthor and friend. I learned a lot about research by working with him. His expertise in norms and communication complexity was very helpful. Special thanks to him for introducing me to dropbox and making me realize the importance of coding. I would like to thank Alantha Newman for working on graph isomorphism and SDPs with me. The long discussions about academics and life in general were really enjoyable. She taught me a lot about research.

The summer internships in NEC and Microsoft Research India were very helpful. I thank Neeraj Kayal and Satya Lokam for being a great host in Bangalore. At NEC, Hari Krovi and Martin Roetteler introduced me to wealth of information about hidden subgroup problem. I specially thank Martin Roetteler for being on my committee and more importantly for his suggestions on this thesis. I also thank Ashwin Nayak for inviting me to Waterloo and being a great host.

The theory students at Rutgers made the experience memorable and fun. The lunch discussions with Devendra Desai, Fengming Wang, Mangesh Gupte and Pravin Shankar were really helpful and highly enjoyable. The trips to various conferences with Dev, Fengming, Luke Friedman, Mangesh and Nikos Leonardos were always filled with fun. Thanks to Carol DiFrancesco and other staff at Rutgers for solving all of my administrative problems.

I would also like to thank countless friends who have provided unconditional support and have been an integral part of my PhD experience. My cricket, football and squash teams cleared my mind from time to time and helped me focus on my academics better. The lunch group at BCC gave an ideal venue for getting fresh again for work. I thank Kirit, Dev, Siddharth, Deepti, Rajesh, Shivangi, Swapnil, Rohan, Varsha, Meenaskshi, Pravin, Madhur, Hari, Viral, Prathima, Shweta and Mangesh from Rutgers for always being there. I also thank "we", my wingmates, Garima and Vikas for their support.

Dedication

To my parents.

Brother-in-law, sister, Bhavya and Kavya.

Brother and sister-in law.

Table of Contents

List of Figures

Chapter 1

Introduction

In many diverse areas of mathematics, problems are studied where an objective function needs to be minimized or maximized while simultaneously satisfying a set of constraints. These problems are called optimization problems, which can be classified depending upon the kind of objective function and the kind of constraints used. One of the simplest class of optimization problems is called linear programming, where both the objective function and the set of constraints are linear. It is heavily used in discrete mathematics, in combinatorial optimization, and in complexity theory. This utility of linear programming derives not only from the fact that large number of problems can be described as linear programs, but also because of our ability to solve these programs efficiently.

The paradigm of *semidefinite programming* is a generalization of linear programming, which can also be solved efficiently and has the power to express more mathematical problems. Semidefinite programs (SDPs) can express any problem written in linear programming and much more. On the other hand, the progress in efficiently solving SDPs was initially slow. This is because the feasible region is not polyhedral, hence, the simplex method[1] could not be applied. But with the advent of more efficient algorithms like the interior point method [Kar84], we have efficient polynomial time algorithms to solve semidefinite optimization problems. These algorithms have made SDP an essential tool in various disciplines of mathematics and computer science.

In theoretical computer science many problems are expressible using semidefinite programs. Then there are other problems which can be approximated well by semidefinite programming relaxations. The idea is to write these problems as integer programs

[1]Simplex algorithm is a popular algorithm for numerically solving linear programming.

(where variables are constrained to be integers); They can then relaxed to be vectors. It turns out that for many of those problems, a semidefinite relaxation approximates the optimum much better than a linear program [Lov, GW95]. Given its efficient algorithms, semidefinite programming is emerging as a major tool for constructing approximation algorithms and for proving bounds on the complexity of underlying problem.

One of the first applications of SDPs in discrete mathematics was the Lovász theta number [Lov79]. It showed that a quantity called *Shannon capacity* can be approximated by an SDP. The other major breakthrough came in 1995 when Goemans and Williamson [GW95] showed an approximation algorithm for the problem *max cut* in combinatorial optimization. It is still the best known algorithm in terms of approximation guarantee. In 2007, Khot et. al. [KKMO07] proved that it is optimal. Recently, another proof of its optimality was given by Raghavendra [Rag08] assuming the unique games conjecture, again using SDPs. The result is much more general and states that for a class of constraint satisfaction problems, SDPs provide the best approximation factor, assuming unique games conjecture.

Semidefinite programming has been really effective in quantum computing. It has provided much closer approximations and sometimes exact formulation in the quantum world. One of the early use of semidefinite programming was to give optimal POVM conditions for distinguishing a set of quantum states. Holevo and Yuen et.al. showed that this can be characterized by an SDP [Hol73, YKL75]. SDPs were also used in proving that $QIP \subset EXP$ [KW00], and now in a seminal result Jain et.al. proved $QIP = PSPACE$ [JJUW10]. Barnum and Knill showed how to reverse quantum dynamics while preserving quantum and classical fidelity [BK00]. All these results used SDPs to characterize various quantities in quantum computing.

Cleve et. al. [CSUU07] showed that the value of certain games, called quantum nonlocal XOR games, can be achieved exactly using SDPs. Later Kempe et.al. [KRT07] showed that SDPs provides a much closer approximation to the value of wider class of games, called nonlocal unique games. In the area of quantum query complexity, Barnum et.al. provided a mathematical representation of quantum query complexity as an SDP feasibility problem [BSS03]. A long series of work showed that adversary methods

give lower bound to quantum query complexity [HNS02, Amb06, Zha05, BSS03, LM04, HLŠ07]. Recently, Ben Reichardt [Rei09, Rei10a] showed that this lower bound is tight for boolean functions.

To summarize, semidefinite programming has been a very useful tool in complexity theory, approximation algorithms and quantum computing. We pursue this line of research further, and show its application in product rules and quantum query complexity. The main results of this thesis are the application of SDPs in these two fields.

Linear programming and semidefinite programming are examples of convex programming, where the objective function and set of constraints are convex. There is a hope to use other convex programming classes in complexity theory. Briefly, in the end, we talk about one such application of *copositive optimization* to complexity theory.

1.1 Applications

As mentioned above, the major part of this thesis deals with the application of SDP in product rule and quantum query complexity. Below we give the motivation behind studying these areas and state the contribution made by this research.

1.1.1 Product rules

A fundamental question in complexity theory is, how the resources needed to compute k copies of a function scale with the resources needed to compute a single instance of the function. Although not always the case, one generally expects and wishes to show that the resources needed grow linearly with k and/or that the success probability decreases exponentially with k. Answers to these questions are known as direct sum theorems, (strong) direct product theorems, or XOR lemmas. It depends on the behavior of the resource, success probability parameters, and whether one considers to output a k-tuple of answers or simply their XOR.

In this thesis, we study a particular approach to proving such product theorems based on semidefinite programming. In this approach, a semidefinite program is developed that approximates the quantity of interest. Then one shows that the semidefinite

program obeys a product rule, this bounds the behavior of the original quantity under product as well. This approach has been successfully used many times in the literature, for example [Lov79, FL92, KKN95, CSUU07, LSŠ08]. Despite this, there has been no general theory to explain when semidefinite programs obey a product rule and when they do not—each of these works had to prove their product theorem from scratch.

We attempt to develop a general theory to explain when the optimum of a semidefinite program is multiplicative under a naturally defined product operation. We find sufficient conditions for the product rule to hold, and also discuss why in some cases it does not hold (Section 3.3). We come up with a necessary condition too (Section 3.6). Our framework is general enough to explain all the semidefinite product theorems in the literature.

There are slight modification to the definition of being "multiplicative". We discuss how our results can be extended to those definitions. The results of this section are published in [MS07, LM08].

1.1.2 Quantum query complexity

There are various complexity measures like certificate complexity and decision tree complexity for functions. One of these measures for functions, is the number of bits needed to compute the function in the worst case, called query complexity. Currently, there are two main ways to get a lower bound on quantum query complexity. One of them, called the adversary method, was developed initially by Bennett et. al. [BBBV97] and then later by Ambainis [Amb02]. It is known that the bound achieved by adversary method for a function can be characterized using a semidefinite program.

In 2007, Hoyer et. al. [HLŠ07] introduced the generalized adversary bound (or negative adversary bound) that lower-bounds the number of input queries needed by a quantum algorithm to evaluate a function [HLŠ07]. This can also be represented as a SDP. This SDP value is known to be tight up to constant factors for functions (total or partial) with boolean output and binary input alphabet [Rei09, Rei10a, Rei10b, Rei10c]. We show that the general adversary bound is tight for any function whatsoever, i.e., with potentially non-boolean input or output alphabets (Theorem 4.0.1). We also show

that quantum query complexity (Q) exhibits a remarkable composition property:

$$Q\big(f(g(x^1),\ldots,g(x^n))\big) = O\big(Q(f)Q(g)\big),$$

for any compatible functions f, g (Corollary 4.5.6). This was previously known only in the boolean case.

Both of these results are obtained by defining a new, but closely related SDP, that we call the witness size. The minimization formulation of the witness size adds more constraints compared to the general adversary bound. This enables dual solutions to correspond to eigenvalue-zero eigenvectors of certain graphs. While the witness size can be strictly larger than the adversary bound, we show that it can be at most a factor of two larger. These results are published in [LMRS10].

1.1.3 Other optimization problems

Instead of optimizing over a semidefinite cone, one can take a look at optimizing over other kinds of convex cones. A natural question is, are there applications in theoretical computer science where these cones can be helpful? We show that one of the recent parallel repetition result [BHH+08] can be interpreted as an application of optimization over the "copositive" cone.

It is known that we can represent NP-hard problems like independent set using a copositive program. It makes this copositive optimization NP-hard to achieve. So this rules out the possibility of using them to obtain approximation algorithms for combinatorial quantities. Still they can be used to give bounds on these quantities. We discuss how copositive programs can be used to give bounds on the value of certain kind of nonlocal games, called unique games. It can be shown that the value of these games, when played multiple times in parallel, approaches the value of the semidefinite relaxation. These ideas are simplification of the article [BHH+08].

1.2 Thesis layout

This thesis is mainly composed of application of SDP in the two domains, product rules and quantum query complexity. There is also a little discussion about how other forms

of convex optimization can be helpful in complexity theory.

In Chapter 2, we give a short introduction to the field of semidefinite programming. The standard form of an SDP as well as other various equivalent formulations are discussed. The chapter ends with explaining how SDPs are mostly used in applications to theoretical computer science, i.e., relaxations and rounding procedures.

Chapter 3 is concerned with product rules derived using semidefinite programming. As discussed in Section 1.1.1, we give sufficient conditions and necessary conditions for a semidefinite program to be multiplicative. These conditions can be used to obtain new product rules. Various product rules from different fields which follow from this general framework are also discussed.

Chapter 4 gives the characterization of quantum query complexity (or negative adversary bound) using SDP. This is done using a new SDP formulation called "witness size", as mentioned in Section 1.1.2. First we give these SDPs for negative adversary bound and witness size. The chapter later discusses the algorithm, which shows that these bounds are tight for quantum query complexity. In the end, we discuss composition results for quantum query complexity.

In Chapter 5, we briefly discuss how copositive programming can be helpful to complexity theory. Then the thesis is concluded by summarizing the results and stating open problems in this field.

Chapter 2

Semidefinite programming

Semidefinite programming is a class of optimization problems over matrix variables. It can be thought of as a generalization of linear programming. In linear programming, we have a linear objective function and linear constraints. An SDP also has a linear objective function, considering entries of the matrix as variables. It can have linear constraints, but it also allows the constraint that a certain matrix is in the semidefinite cone. This is defined formally in Section 2.3.

2.1 Notation

For a natural number $n \in \mathbf{N}$, let $[n] = \{1, 2, \ldots, n\}$. Upper case letters will be used to denote matrices and lower case letters vectors. Bold capital letters will be used for tuples of matrices, e.g., $\mathbf{A} = (A^{(1)}, \ldots, A^{(m)})$. The multiplication between vector y and tuple of matrices $\mathbf{A}$ is defined as $y^T\mathbf{A} = \sum_{i=1}^{m} y_i A^{(i)}$.

For two matrices A, B of the same size, $A \circ B$ denotes their entry-wise product, also known as the Hadamard or Schur product. The summation of all the entries in the matrix $A \circ B$ is denoted by $A \bullet B$ (dot product of A and B), and defined as

$$A \bullet B = \sum_{i,j} A_{i,j} B_{i,j}. \tag{2.1.1}$$

The inner product between two vectors u and v will be denoted by $u^T v$ or $\langle u|v\rangle$.

If not specified, we will assume all matrices to be symmetric. An optimization problem can be denoted by any one of the symbols $\pi, \sigma, \pi(X), \sigma(X)$. Here X is the variable in the optimization problem[1]. Given a primal instance π, the dual of that instance will be denoted by π^*.

[1]The variable X can be dropped from the notation (π, σ), when it is clear from the context.

Any optimization problem has an objective function, the function we are trying to minimize or maximize. It has a feasible region (the set of admissible X's), defined by the constraints of the problem. So the task is to maximize/minimize the objective function dependent on X, under the condition that X belongs to the feasible region.

2.2 Positive semidefinite matrices

An $n \times n$ symmetric matrix M is called a positive semidefinite matrix if, for all vectors $v \in \mathbb{R}^n$, the quadratic form $v^T M v \geq 0$. It is called positive definite, if $v^T M v > 0$. When a matrix M is positive semidefinite, it is denoted by symbol $M \succeq 0$ ($M \succ 0$, when it is positive definite). There are other equivalent definitions also. We state them without proof [HJ85].

Theorem 2.2.1. *The following are equivalent for an $n \times n$ symmetric matrix $M \in \mathbb{R}^{n \times n}$.*

1. *M is positive semidefinite, i.e., $\forall\ v \in \mathbb{R}^n,\ v^T M v \geq 0$.*
2. *All eigenvalues of M are nonnegative.*
3. *M is the Gram matrix of a set of vectors $x_1, \cdots, x_n \in \mathbb{R}^k$, for some k. So, for all $i, j \in [n]$, $M_{i,j} = \langle x_i | x_j \rangle$.*
4. *For some vectors $v_1, \cdots, v_k \in \mathbb{R}^n$, $M = v_1 v_1^T + \cdots + v_k v_k^T$.*

The set of positive semidefinite matrices is a "cone". This means that for any two positive semidefinite matrices M_1, M_2, given any $\alpha, \beta \geq 0$, it holds that $\alpha M_1 + \beta M_2 \succeq 0$.

The dual cone C^* of a cone C is the set of vectors whose dot product with any element of the cone C is non-negative, i.e.,

$$C^* = \{y \in \mathbb{R}^n : y^T x \geq 0\ \forall x \in C\}. \tag{2.2.1}$$

It is known that the dual of the semidefinite cone is the semidefinite cone itself. This implies, for any two positive semidefinite matrices M_1, M_2, we know $M_1 \bullet M_2 \geq 0$ (even $M_1 \circ M_2 \succeq 0$). On the other hand, given a K, s.t., $\forall M \succeq 0 K \bullet M \geq 0$, then K is positive semidefinite [BV04].

2.3 Affine semidefinite program instances

We take a semidefinite program to be described by triples $\pi = (J, \mathbf{A}, b)$, where

- J is the objective matrix, a symmetric matrix of dimension $n \times n$,
- $\mathbf{A} = (A^{(1)}, \ldots, A^{(m)})$ is a list of m symmetric matrices, each of dimension $n \times n$. $\mathbf{A}$ describes a transformation from $\mathbf{R}^{n \times n} \to \mathbf{R}^m$ where $\mathbf{A}(X) = (A^{(1)} \bullet X, \ldots, A^{(m)} \bullet X)$,
- b is a vector of length m.

With π we associate a semidefinite programming instance in the following standard form with optimal value $\alpha(\pi)$:

$$\begin{aligned} \max_X \quad & J \bullet X \\ \text{s.t.} \quad & \mathbf{A}(X) = b \\ & X \succeq 0. \end{aligned} \tag{2.3.1}$$

We define the dimension of the instance to be (n, m). This semidefinite programming instance is called affine as it only has equality constraints. Note, however, that this formulation is completely general as one can encode inequality constraints by introducing slack variables as necessary and enforcing non-negativity constraints by putting them on the diagonal of X.

Notice that the objective function $J \bullet X$ is linear in the entries of the matrix X. There are linear constraints of the form $A^{(i)} \bullet X = b_i$. The semidefinite condition is enforced by the constraint, that matrix X lies in the semidefinite cone ($X \succeq 0$). These constraints can also be thought of as, "feasible region is the intersection of a polyhedra and the semidefinite cone".

2.3.1 Duality

Like linear programming, duality theory is an important part of semidefinite programming. The "dual" of a semidefinite program is another semidefinite optimization problem (SDP). It has the opposite objective, minimization if original problem is maximization and vice versa. The original problem is called "the primal". The dual of the dual is the primal [BV04].

We will need the dual of π (defined by Eq. 2.3.1), which we denote by π^*. For the method to express the dual see, for example, [BV04, Ali95, Lov]. The dual is defined as

$$\begin{aligned} \min_{y} \quad & y^T b \\ \text{s.t.} \quad & y^T \mathbf{A} - J \succeq 0. \end{aligned}$$

where y is a column vector of length m. Here, $y^T\mathbf{A}$ is the matrix $\sum_{k=1}^{m} y_k A^{(k)}$.

Without loss of generality assume that our primal is a maximization problem (like Eq. 2.3.1). The theorem of weak duality states, for any feasible solutions X, y of primal π and dual π^* respectively,

$$\pi(X) \leq \pi^*(y).$$

Hence the value of the dual is an upper bound on the value of the primal. Similarly the value of the primal is a lower bound on the value of dual. The situation in which these two optimal values are equal ($\alpha(\pi) = \alpha(\pi^*)$) is known as strong duality.

In the case of semidefinite programs, the value of the primal is not always equal to the value of the dual. In other words, strong duality is not necessarily true. But it follows if either the primal or the dual is strictly feasible, i.e., if there exist a feasible $X \succ 0$ for the primal or there is a y such that $y^T\mathbf{A} - J \succ 0$ for the dual, then strong duality is achieved. In most of the cases, and all the cases discussed in this thesis, this condition holds. Hence we assume strong duality in the complete thesis. More information about strong duality can be found in the excellent surveys [BV04, Ali95].

2.3.2 Vector formulation

For applications in complexity theory it turns out that another formulation of SDPs where vectors are the variables (instead of the matrices) is used more often. From Theorem 2.2.1 any $n \times n$ positive semidefinite matrix M can be thought of as the Gram matrix of vectors $v_1, v_2 \cdots, v_n$, i.e., the i, j^{th} entry of M is equal to the inner product ($\langle v_i | v_j \rangle$) between v_i and v_j. Conversely, any Gram matrix of n vectors is a positive semidefinite matrix.

This equivalence gives us another way to express the semidefinite programming problem π mentioned above.

$$\max_{v_i} \quad \sum_{i,j} J_{i,j} \langle v_i | v_j \rangle \tag{2.3.2}$$

$$\forall k \quad \sum_{i,j} A^k_{i,j} \langle v_i | v_j \rangle = b_k \tag{2.3.3}$$

Notice here that there is no positive semidefinite cone condition, because if every i, j^{th} entry of the matrix can be represented as $\langle v_i | v_j \rangle$, then it is positive semidefinite.

2.4 SDPs as relaxations

Semidefinite programs are a convenient way to estimate many combinatorial quantities. Sometimes they express the *quantity of interest* exactly [CSUU07], and in other cases they approximate the quantity tightly [GW95, ARV04, KRT07, Rei09]. The concept of relaxation is very important for all these applications.

These relaxations help in approximating a quantity. The quantity is first expressed exactly as an integer program or a convex optimization problem with some non-convex constraints. These programs cannot be solved efficiently and in most of the cases are NP-complete.

The idea is to "relax" the program to an SDP which can be solved efficiently. A "relaxation" $\bar{\sigma}(X)$ of an optimization problem $\sigma(X)$ (X is the variable in the optimization problem), is another optimization problem. Any feasible solution X of problem $\sigma(X)$ should be a solution of $\bar{\sigma}(X)$ with the same objective value. This shows that the value of $\bar{\sigma}(X)$ is a lower bound (upper bound) if $\sigma(X)$ is a minimization (maximization)

problem. A relaxation, by definition, is a bound on the original quantity of interest. It is useful if it can be efficiently computed or is related to some other quantity of interest.

So the approach is to relax an integer program/non-convex program into a semidefinite program, which we know is efficiently computable. In the case of integer programs, we can drop the constraint that variables are integer and assume them to be vectors. This gives us a semidefinite program (Section 2.3.2), if the integer program has linear constraints and objective function. That is one of the reasons why vector formulation is so important in complexity theory.

But solving a relaxation is not sufficient. We need to show that the relaxation is close to the original quantity of interest. This is achieved by showing that the optimal solution of a relaxation can be converted into a solution of the original non-convex optimization problem without losing much in the objective value. For the integer programming case, we need to convert the vector solution into an integer solution. This is known as "rounding" [GW95, Rag08].

Hence, the process of creating a relaxation and then showing a good rounding algorithm, proves that the semidefinite program is close to the quantity of interest. This can be used to give bounds on the quantity or obtain good approximation algorithms. In the next section, we will discuss few examples that highlight the use of this strategy.

2.4.1 Examples

One of the first applications of semidefinite programs in complexity theory was the approximation algorithm for max-cut given by Goemans and Williamson [GW95]. It is a good example for the relaxation and rounding strategy we outlined above. In the max-cut problem, we are given a graph $G = (V, E)$, and we need to find the cut which has maximum number of edges. The cut is specified by a partition of the vertex set V into two parts. The number of edges in the cut is the number of edges going between the two partitions.

This optimization can be written as an integer program, where variables can only take values from $\{1, -1\}$. The set of vertices which are assigned 1 form one part of the

partition, and the rest another.

$$\max_{y_i} \quad \sum_{(i,j)\in E} \frac{1 - y_i y_j}{2}$$

$$\forall i \in V \quad y_i \in \{1, -1\}$$

This integer program is relaxed to get a semidefinite program. Here the variables for every vertex are allowed to be vectors instead of just being from $\{1, -1\}$.

$$\max_{v_i} \quad \sum_{(i,j)\in E} \frac{1 - \langle v_i | v_j \rangle}{2}$$

$$\forall i \in V \quad v_i \in \mathbb{R}^{|V|}$$

This can be solved in polynomial time using semidefinite programming solvers. Goemans and Williamson showed that the resulting vectors can be converted into integers by a simple rounding procedure. We take a random hyperplane and assign ± 1 to vertex i, depending upon which side of the hyperplane v_i lies on. The expected value of the integer solution is at least .878 times the vector solution. This gives a .878 approximation algorithm for max-cut [GW95].

Taking another example, the maximum/minimum eigenvalue of a matrix can be calculated as a semidefinite program. More generally, suppose a symmetric matrix $A(x)$ depends affinely on $x \in \mathbb{R}^n$, i.e., $A(x) = A_0 + x_1 A_1 \cdots + x_n A_n$. Then the minimization of the maximum eigenvalue of $A(x)$ can be computed using the SDP

$$\begin{aligned} \min_{x} \quad & t \\ s.t. \quad & tI - A(x) \succeq 0 \end{aligned}$$

Here t, x are the variables. This format is same as the dual problem mentioned in the Section 2.3.1.

Chapter 3

Product rules

A prevalent theme in complexity theory is what we might roughly call product theorems. These results look at, how the resources needed to accomplish several independent tasks scale with the resources needed to accomplish the tasks individually. Let us look at a few examples of such questions:

Shannon capacity: Let $\alpha(G)$ be the size of a largest independent set in a graph G. How does $\alpha(G)$ compare with amortized independent set size $\lim_{k\to\infty} \alpha(G^k)^{1/k}$? This last quantity, known as the Shannon capacity, gives the effective alphabet size of a graph where vertices are labeled by letters and edges represent letters which can be confused if adjacent [Lov79].

Hardness amplification: Product theorems naturally arise in the context of hardness amplification. If it is hard to evaluate a function $f(x)$, then an obvious approach to create a harder function is to evaluate two independent copies $f'(x, y) = (f(x), f(y))$ of f. There are different ways that f' can be harder than f—a direct sum theorem aims to show that evaluation of f' requires twice as many resources as needed to evaluate f; direct product theorems aim to show that the error probability to compute f' is larger than that of f, given the scaled up amount of resources [JKS10].

Soundness amplification: Closely related to hardness amplification is what we might call soundness amplification. This arises in the context of interactive proofs where one wants to reduce the error probability of a protocol by running several checks in parallel. The celebrated parallel repetition theorem shows that the soundness of multiple prover interactive proof systems can be boosted in this manner [Raz98].

3.1 Semidefinite programming approach to product theorems

The previous examples illustrate that many important problems in complexity theory have dealt with product theorems. One successful approach to these types of questions has been through semidefinite programming. The use of semidefinite programming has proliferated in many areas of theoretical computer science, for example in combinatorial optimization [GW95, KMS98, ARV04], quantum computing [BSS03, CHTW04, CSUU07, HLŠ07], and complexity theory [FL92, LLS06, LSŠ08], and it has also proven useful to show product theorems.

In this approach, we want to know how some quantity $\sigma(G)$ behaves under the product operation. So we take a look at the semidefinite approximation/relaxation $\bar{\sigma}(G)$ of $\sigma(G)$. Then the hope is to show

1. $\bar{\sigma}(G)$ provides a good approximation to $\sigma(G)$ (rounding).

2. $\bar{\sigma}(G)$ obeys a product theorem $\bar{\sigma}(G \times G) = \bar{\sigma}(G)\bar{\sigma}(G)$.

So we obtain that the original quantity $\sigma(G)$ must approximately obey a product rule as well.

It is shown that all the semidefinite programs do not multiply. This strategy gives a very natural approach, which shows when a semidefinite program obeys a product theorem. Namely, we use the maximization formulation of the semidefinite program to show that $\bar{\sigma}(G \times G) \geq \bar{\sigma}(G)^2$ by combining optimal solutions for the G instance into a solution for the $G \times G$ instance. Similarly, then we try to show that $\bar{\sigma}(G \times G) \leq \bar{\sigma}(G)^2$ by considering the dual minimization formulation and showing that an optimal dual solution can be combined into a solution for the $G \times G$ instance. Note that in this strategy one does not need to use the fact that the value of semidefinite programs can be computed efficiently.

Let us see how this approach has been used for the above questions.

Shannon capacity: Perhaps the first application of this technique was to the Shannon capacity of a graph G. Lovász developed a semidefinite quantity, the Lovász theta function $\vartheta(G)$. He showed that it is an upper bound on the independence number of a

graph and that $\vartheta(G \times G) = \vartheta(G)^2$. In this way he determined the Shannon capacity of the pentagon, resolving a long-standing open problem [Lov79].

Hardness amplification: Karchmer, Kushilevitz, and Nisan [KKN95] notice that another program introduced by Lovász [Lov75], the fractional cover number, can be used to characterize non-deterministic communication complexity, up to small factors. As this program also perfectly products, they obtain a direct sum theorem for non-deterministic communication complexity.

As another example, Linial and Shraibman [LS08] show that a semidefinite programming quantity γ_2^∞ characterizes the discrepancy method of communication complexity, up to constant factors. Lee, Shraibman and Špalek [LSŠ08] then use this result, together with the fact that γ_2^∞ perfectly products, to show a direct product theorem for discrepancy, resolving an open problem of Shaltiel [Sha03].

Soundness amplification: Although the parallel repetition theorem was eventually proven by other means [Raz98, Hol07], one of the first positive results did use semidefinite programming. Feige and Lovász [FL92] show that the acceptance probability, $\omega(G)$, of a two-prover interactive game can be represented as an integer program. They then relax this to a semidefinite program, $\bar{\omega}(G)$, and show that this quantity perfectly products. In this way, they are able to show that if $\omega(G) < 1$ then $\sup_{k\to\infty} \omega(G^k)^{1/k} < 1$, for a certain class of games G known as unique games.

Recently, there has been a lot of renewed interest in studying parallel repetition through semidefinite programming. Part of this interest stems from the close connection between semidefinite programming and the value of a two-prover interactive game where the provers are allowed to share entanglement. The value of a special kind of such a game, known as an XOR game, can be exactly represented by a semidefinite program [Tsi87, CHTW04]. Cleve et al. [CSUU07] show that this semidefinite program obeys a product theorem to obtain a perfect parallel repetition theorem for these kinds of games. Kempe, Regev, and Toner [KRT07] look at the broader class of unique games with entanglement and use semidefinite programming techniques to get a parallel repetition theorem for these games.

Very recently, Barak et al. [BHH+08] show that semidefinite programming is inherently connected to parallel repetition. Roughly speaking, they show that the amortized value of a unique game repeated in parallel converges to the value of the natural semidefinite relaxation of the unique game studied in the original paper [FL92].

3.1.1 Our contribution

We hope that this selection of examples demonstrates the power and usefulness of the semidefinite programming approach to proving product theorems. Despite these successes, however, we are not aware of any work which systematically investigates the conditions under which semidefinite product theorems hold. This is what we attempt to do in this chapter. We identify a large class of semidefinite programming instances that obey the product rule, and in particular we are able to explain all of the product theorems in the works [Lov79, FL92, CSUU07, LSŠ08, KRT07]. We also discuss some examples where product theorems do not hold, and give a necessary condition for a product theorem to hold, at least if one proceeds by the most natural proof technique.

We think that the sufficiency conditions developed here will already be of use for researchers using the semidefinite programming approach to prove product theorems. Indeed, rather than proving their product theorem from scratch, Kempe et al. [KRT07] were able to appeal to the results in our preliminary article [MS07]. From a mathematical point of view, however, there is still much to be done to reach a full understanding of when semidefinite programs product. We hope to provoke ideas and set the scene for what one day might be a complete classification.

3.2 Product instances

Consider a semidefinite instance given in the standard form, as defined in Section 2.3.

$$\begin{aligned} \max_X \quad & J \bullet X \\ & \mathbf{A}(X) = b \\ & X \succeq 0. \end{aligned}$$

We now define a notion of what it means to take the product of two such semidefinite programming instances.

Definition 3.2.1. *Let $\pi_1 = (J_1, \mathbf{A}_1, b_1)$ and $\pi_2 = (J_2, \mathbf{A}_2, b_2)$ be two affine semidefinite programming instances with dimensions (n_1, m_1) and (n_2, m_2), respectively. We define the product instance as $\pi_1 \times \pi_2 = (J_1 \otimes J_2, \mathbf{A}_1 \otimes \mathbf{A}_2, b_1 \otimes b_2)$, where $\mathbf{A}_1 \otimes \mathbf{A}_2$ is by definition the list $(A_1^{(k)} \otimes A_2^{(l)})_{k,l}$ of length $m_1 m_2$ of $n_1 n_2 \times n_1 n_2$ matrices. The product instance has dimensions $(n_1 n_2, m_1 m_2)$.*

The natural question to ask is if two instances of semidefinite programs behave nicely under this product operation.

Definition 3.2.2. *Two semidefinite programming instances π_1 and π_2 are said to obey a product theorem if and only if $\alpha(\pi_1 \times \pi_2) = \alpha(\pi_1)\alpha(\pi_2)$.*

The rest of this chapter focuses on finding conditions, both sufficient and necessary, for a product theorem to hold.

From a mathematical point of view, this definition of product is the most natural. In applications, however, one generally has a notion of product in mind which is native to the problem at hand, and which might not always agree with that of Definition 3.2.1. It is possible that taking the tensor product of the constraint matrices $\mathbf{A}_1 \otimes \mathbf{A}_2$ creates unintended constraints, or, on the contrary, that one in fact desires more constraints than just those defined by $\mathbf{A}_1 \otimes \mathbf{A}_2$. This is the most subtle aspect of applying our theory in practice. Later in the chapter, Section 3.7 and Section 3.4, we discuss alternative notions of product used in the literature and how they fit into our setting.

3.2.1 Counterexample to the product theorem

In this section we give an example of a semidefinite programming instance where the product theorem does not hold. For a symmetric and square matrix M, let $\lambda_1(M)$ denote the largest eigenvalue of M. The largest eigenvalue of M, in contrast to the similar notion of spectral norm, is not multiplicative. Indeed, let M be a matrix with maximal eigenvalue 1 and minimal eigenvalue -2. Then, using the fact that under

the tensor product the spectra of matrices multiply, we get that $M \otimes M$ has maximal eigenvalue $4 \neq 1^2$ (the corresponding spectral norms would be 2 for M and 4 for $M \otimes M$).

Proposition 3.2.3. *The largest eigenvalue of a symmetric matrix can be formulated as the optimal value of a semidefinite program. This program does not obey a product theorem.*

Proof. First notice that

$$\lambda_1(M) = \min\{\lambda \mid \lambda I - M \succeq 0\}. \tag{3.2.1}$$

This is the formulation of the dual (minimization) instance. Observe that $m = 1, \mathbf{A} = (I), J = M$ and $b = 1$. For the sake of completeness we also describe the primal problem:

$$\lambda_1(M) = \max\{M \bullet X \mid \mathrm{Tr}(X) = 1, \;\; X \succeq 0\}. \tag{3.2.2}$$

The product instance associated with two matrices, M_1 and M_2, has parameters $I = I_1 \otimes I_2$, $M = M_1 \otimes M_2$ and $b = 1$. Since I is the identity matrix of appropriate dimensions, the optimum value of this instance is exactly the maximal eigenvalue of $M_1 \otimes M_2$. On the other hand, as described in the beginning of this section, the maximal eigenvalue program does not obey a product theorem. □

As remarked above, the spectral norm of a symmetric matrix M, that is, the largest eigenvalue of M in magnitude, does obey a product theorem and is described by a very similar semidefinite program:

$$\|M\| = \max\{|M \bullet X| \mid \mathrm{Tr}X = 1, \;\; X \succeq 0\}. \tag{3.2.3}$$

The reader might think that a product theorem can always be rescued by taking the absolute value of the objective function, Next we give a counterexample to this conjecture. Consider a very simple linear program (it is also a semidefinite program):

$$\begin{aligned} &\max x_1 - x_2 \\ &x_1 - x_2 + x_3 = 2 \\ &x \geq 0. \end{aligned}$$

Here x is the vector $(x_1, x_2, \cdots, x_n)$. Clearly the value of this program is 2; since x_3 is non-negative. Now if we take the product of this program with itself

$$\begin{aligned} &\max x_{1,1} - x_{1,2} - x_{2,1} + x_{2,2} \\ &x_{1,1} - x_{1,2} - x_{2,1} + x_{2,2} + x_{3,1} + x_{1,3} + x_{3,3} - x_{2,3} - x_{3,2} = 4 \\ &x \geq 0. \end{aligned}$$

Now, the value of the objective function can be raised indefinitely by raising the value of $x_{2,3}$. So the value of the program is not 4. The product program would have been the same, if we had taken the product in terms of semidefinite representation. So taking the absolute value does not save the product theorem.

3.2.2 The product solution

In Section 3.2.1 we saw an example of an affine semidefinite program that does not obey a product rule. Therefore, for the product rule to hold we need to look for proper subclasses of all affine instances.

Let π_1 and π_2 be two affine instances, with optimal solutions X_1 and X_2 for the primal, and optimal solutions y_1 and y_2 for the dual. For our entire discussion, we will assume that so-called strong duality holds for π_1 and π_2 (see [BV04, Ali95], Section 2.3.1). This implies that $\alpha(\pi_1) = \alpha(\pi_1^*)$, i.e., the primal and dual values agree, and similarly for π_2 and $\pi_1 \otimes \pi_2$.

The first instinct for proving the product theorem would be to show if X_1, X_2 are optimal solutions to π_1, π_2 respectively, then $X_1 \otimes X_2$ is a solution of the product instance. This solution $X_1 \otimes X_2$ has objective value $\alpha(\pi_1)\alpha(\pi_2)$, which gives a lower bound on $\alpha(\pi_1 \times \pi_2)$. Similarly, if y_1, y_2 are optimal solutions to π_1^*, π_2^*; then $y_1 \otimes y_2$ is a solution of the dual of the product instance. This will give the lower bound of same value $\alpha(\pi_1)\alpha(\pi_2)$. The above two potential solutions for the product instance and its dual we call the *product-solution* and the *dual product-solution*. In other words, in order to show that the product rule holds for π_1 and π_2 it is sufficient to prove:

1. Feasibility of the product-solution: $(\mathbf{A}_1 \otimes \mathbf{A}_2)(X_1 \otimes X_2) = b_1 \otimes b_2$;

2. Feasibility of the dual product-solution: $(y_1 \otimes y_2)^T(\mathbf{A}_1 \otimes \mathbf{A}_2) - J_1 \otimes J_2 \succeq 0$;

3. Objective value of the primal product-solution: $(J_1 \otimes J_2) \bullet (X_1 \otimes X_2) = (J_1 \bullet X_1)(J_2 \bullet X_2)$;

4. Objective value of the dual product-solution: $(y_1 \otimes y_2)^T(b_1 \otimes b_2) = (y_1^T b_1)(y_2^T b_2)$.

We also need that $X_1 \otimes X_2 \succeq 0$, but this is automatic as $X_1, X_2 \succeq 0$. Which of 1–4 fail to hold in general? Basic linear algebra gives that conditions 1, 3 and 4 hold without any further assumption. This immediately gives

Proposition 3.2.4. *Let π_1 and π_2 be two affine instances. Then $\alpha(\pi_1 \times \pi_2) \geq \alpha(\pi_1)\alpha(\pi_2)$.*

3.3 Sufficient conditions for a product theorem

From the previous section, any property of $(\mathbf{A}, J, b)$ which implies Condition 2 (dual feasibility) will be a sufficient condition for the product theorem to hold. In this section we present some sufficient conditions for dual feasibility.

The condition for dual feasibility is $(y_1 \otimes y_2)^T(\mathbf{A}_1 \otimes \mathbf{A}_2) - J_1 \otimes J_2 \succeq 0$. We already know that $y_1^T \mathbf{A}_1 - J_1 \succeq 0$ and $y_2^T \mathbf{A}_2 - J_2 \succeq 0$. Note that it is not true in general that $A \succeq B$ and $C \succeq D$ implies $A \otimes C \succeq B \otimes D$. One can take the very simple counterexample with scalars where $A, C = 1$ and $B, D = -2$

Let us assume, however, that $y_1^T \mathbf{A}_1 + J_1$ and $y_2^T \mathbf{A}_2 + J_2$ are also positive semidefinite. Then

$$(y_1^T \mathbf{A}_1 - J_1) \otimes (y_2^T \mathbf{A}_2 + J_2) = y_1^T \mathbf{A}_1 \otimes y_2^T \mathbf{A}_2 - J_1 \otimes y_2^T \mathbf{A}_2 + y_1^T \mathbf{A}_1 \otimes J_2 - J_1 \otimes J_2 \succeq 0. \quad (3.3.1)$$

Also

$$(y_1^T \mathbf{A}_1 + J_1) \otimes (y_2^T \mathbf{A}_2 - J_2) = y_1^T \mathbf{A}_1 \otimes y_2^T \mathbf{A}_2 - y_1^T \mathbf{A}_1 \otimes J_2 + J_1 \otimes y_2^T \mathbf{A}_2 - J_1 \otimes J_2 \succeq 0. \quad (3.3.2)$$

Taking the average of the right hand sides of Equations (3.3.1) and (3.3.2) we obtain

that

$$y_1^T \mathbf{A}_1 \otimes y_2^T \mathbf{A}_2 - J_1 \otimes J_2 \succeq 0.$$
$$\Rightarrow (y_1 \otimes y_2)^T (\mathbf{A}_1 \otimes \mathbf{A}_2) - J_1 \otimes J_2 \succeq 0.$$

which is the desired Condition 2.

Note that in this proof, we only assumed the positivity of $y_c^T A_c + J_c \succeq 0$. That gives us our first sufficient condition.

Lemma 3.3.1. *Let π_1, π_2 be semidefinite programs for which strong duality holds. Let y_1, y_2 be optimal solutions to the dual formulations π_1^*, π_2^* respectively. If $y_c^T \mathbf{A}_c + J_c \succeq 0$ for both $c \in \{1, 2\}$, then the product theorem holds for π_1 and π_2.*

Unfortunately, this condition depends on an optimal dual solution of the semidefinite instance, something that we generally will not know. Now we will derive two less general sufficient conditions that have the advantage of only depending upon $J_c, \mathbf{A}_c, b_c$ for $c \in \{1, 2\}$, the parameters of our semidefinite instance.

3.3.1 Positivity of the matrix J

We saw above a counterexample to the statement that $A \succeq B$ and $C \succeq D$ implies $A \otimes C \succeq B \otimes D$. In the case of scalars, however, if $B, D \geq 0$, then the implication is true. Our first simple condition is based on the analogous fact that if $B, D \succeq 0$ then the implication also holds.

Theorem 3.3.2. *Assume that both J_1 and J_2 are positive semidefinite. Then $\alpha(\pi_1 \times \pi_2) = \alpha(\pi_1)\alpha(\pi_2)$.*

Proof. As we noted in Section 3.2.2 it is sufficient to show that Condition 2 of that section holds. By our assumptions on y_1 and y_2 we have that $y_1^T \mathbf{A}_1 - J_1$ and $y_2^T \mathbf{A}_2 - J_2$ are positive semi-definite. So $y_1^T \mathbf{A}_1 + J_1$ and $y_2^T \mathbf{A}_2 + J_2$ are also positive semi-definite, since they arise as sums of two positive matrices ($y_1^T \mathbf{A}_1 + J_1 = (y_1^T \mathbf{A}_1 - J_1) + 2J_1$). Hence by Lemma 3.3.1, condition 2 is satisfied. □

The Lovász theta number [Lov79] is an example that falls into this category. Consider the definition of Lovász theta number in [Sze94]. Here E denotes the all ones matrix.

$$\begin{aligned} \max_X \quad & E \bullet X \\ & \mathrm{Tr}(X) = 1 \\ & X_{i,j} = 0 \text{ for all edges } (i,j) \\ & X \succeq 0. \end{aligned}$$

The objective matrix is the all ones matrix and so is positive semidefinite. The objective matrix remains positive semidefinite even if we consider the weighted version of the theta number [Knu94]. In the weighted case, J is of the form ww^T for some column vector w.

3.3.2 All $A^{(k)}$ are block diagonal, and J is block anti-diagonal

We now look for other situations where the assumption of Lemma Lemma 3.3.1 holds. Following the lead of Cleve et al. [CSUU07], we can identify another sufficient condition which arises surprisingly often in practice.

For this condition, we need to introduce the notions of what we call "block diagonal" and "block anti-diagonal" matrices. An n-by-n matrix A is block diagonal, if we can partition $[n]$ into two sets (C_1, C_2), such that, $A[i,j] = 0$ whenever i, j do not lie in the same set. Similarly we say that A is block anti-diagonal, if there is a partition of $[n]$ into two sets (C_1, C_2), such that, $A[i,j] = 0$ whenever i, j lie in the same set. Pictorially, these matrices look as follows:

Block anti-diagonal Block diagonal

$$\begin{pmatrix} 0 & Q \\ Q^T & 0 \end{pmatrix} \qquad \begin{pmatrix} P & 0 \\ 0 & Q \end{pmatrix}$$

Lemma 3.3.3. *Let A, B be n-by-n symmetric matrices such that A is block diagonal and B is block anti-diagonal with respect to the same partition of $[n]$. If $A - B \succeq 0$ then $A + B \succeq 0$.*

Proof. Let $C_1 \uplus C_2 = [n]$ be a partition for which A is block diagonal and B is block anti-diagonal. Define the diagonal matrix D as $D[i,i] = 1$ if $i \in C_1$ and $D[i,i] = -1$ if $i \in C_2$. Note that D is an orthogonal matrix and $D^T(A - B)D = A + B$. Thus $A - B$ and $A + B$ have the same spectra. Hence one is positive semidefinite if and only if the other is. □

Now consider all the matrices $A^{(k)}$ to be block-diagonal and J to be block anti-diagonal. For those instances, from Lemma 3.3.1 and Lemma 3.3.3, product rule will hold. We can generalize it further for case when J is of the form $J_1 + J_2$, where J_1 is of the form as before and J_2 is positive semidefinite ($y^T\mathbf{A}$ should still be block diagonal). Notice that the block diagonality of $y^T\mathbf{A}$ automatically holds if $\mathbf{A} = (A^{(1)}, \ldots, A^{(m)})$, where all $A^{(k)}$ are block diagonal with respect to the same partition. We summarize the findings of this section in the following theorem:

Theorem 3.3.4. *Let $\pi_c = (\mathbf{A}_c, J_c, b_c)$ for both $c \in \{1,2\}$ be affine instances for which strong duality holds and such that:*

1. *$\mathbf{A}_c = (A_c^{(1)}, \ldots, A_c^{(m)})$, where each $A_c^{(k)}$ is block diagonal.*
2. *$J_c = J_c' + J_c''$ where J_c' is block anti-diagonal and $J_c'' \succeq 0$ for $c \in \{1,2\}$.*

All $A_1^{(i)}(J_1')$ matrices are block diagonal (antidiagonal) with respect to the same partition and similarly for $A_2^{(i)}(J_2')$. Then the product theorem holds for π_1 and π_2.

Proof. Let y_1, y_2 be optimal solutions for π_1^*, π_2^* respectively. From the feasibility of y_c and $J_c'' \succeq 0$, we have $y_c^T\mathbf{A}_c - J_c' \succeq 0$. Also Notice that $y_c^T\mathbf{A}_c$ is block diagonal because each $A_c^{(i)}$ is block diagonal. Thus we can apply Lemma 3.3.3 to obtain $y_c^T\mathbf{A}_c + J_c' \succeq 0$. Finally, this implies $y_c^T\mathbf{A}_c + J_c \succeq 0$ and we obtain the theorem by applying Lemma 3.3.1. □

The product theorem of Cleve et al. [CSUU07] for the quantum value of XOR games falls under this condition. The SDP for the quantum value of a 2-prover nonlocal XOR game can be written as:

$$\begin{aligned} \max \begin{pmatrix} 0 & \frac{1}{2}A \\ \frac{1}{2}A^T & 0 \end{pmatrix} \bullet X \\ \text{s.t.} \quad diag(X) = e \\ X \succeq 0 \end{aligned}$$

Here A is some cost matrix, but the whole objective matrix is anti block diagonal. The constraints are only on diagonal (so $\mathbf{A}$ is block diagonal). Note that the product operation defined in [CSUU07] is not a true tensor product as we use here, but rather another kind of product, called "bipartite tensor product". It is relatively straightforward, however, to see that the value of the programs under these two notions of product agree and this is nicely explained in [KRT07]. For the sake of completeness, in Appendix A, we include the definition and show that our product Theorem 3.3.4 implies product theorem for "bipartite tensor product".

3.3.3 A generalized condition

Notice the condition given in Theorem Theorem 3.3.4. One seemingly unnatural property of the condition that $\mathbf{A}$ is block diagonal and J is block anti-diagonal, however, is that it depends on the basis in which $\mathbf{A}$ and J are presented. On the other hand, obeying a product theorem intuitively seems like a basis independent property. This intuition is difficult to formalize in our framework as even the property of being positive semidefinite depends on the basis—while A and SAS^{-1} have the same eigenvalues, SAS^{-1} is not necessarily symmetric if A is. Nonetheless, this line of thinking motivates the following generalization:

Lemma 3.3.5. *Suppose that $A - B \succeq 0$. If there exists a matrix S such that*

- $SAS^T = A$
- $B + SBS^T \succeq 0$.

Then $A + B \succeq 0$.

First let us see how this generalizes our previous two conditions. To obtain the case where J is positive semidefinite (Theorem 3.3.2), we may simply take S to be the identity matrix. For the case where the matrices $\mathbf{A}$ are block diagonal and J is block antidiagonal with respect to the same partition (Theorem 3.3.4), we may take S to be defined as the diagonal matrix D in the proof of Lemma 3.3.3. In this case $J + SJS^T = 0$.

Now we prove the lemma.

Proof. By assumption $A - B \succeq 0$, so let $XX^T = A - B$ be a factorization. Now consider $S(A - B)S^T$. This matrix is also positive semidefinite as it can be factored as $SX(SX)^T$. Using now the assumption that $SAS^T = A$ we have $A - SBS^T \succeq 0$. Finally, if $B + SBS^T \succeq 0$ then we can add this to $A - SBS^T$ to obtain the conclusion of the lemma. □

We summarize the sufficient conditions given in this section through the figure below (Figure 3.1). Every node in the tree is a sufficient condition in itself. Every children is derived from the parent condition. The root node in the tree (Lemma 3.3.1) is the most basic and strongest among the four. This condition implies all the others as they all take this approach to showing the product theorem. Next is the condition from Lemma 3.3.5. The two leaves can be derived from this node. It is an interesting open question to find other conditions which can be derived from this Lemma 3.3.5.

3.4 Nonnegativity constraints

Two examples in the literature, the product theorem of Feige and Lovász [FL92] for a semidefinite relaxation of the value of interactive games and the product theorem of [LSŠ08] for the discrepancy bound in communication complexity, still do not fit into the framework we have developed. The reason is that these programs do not only contain equality constraints, but also nonnegativity constraints. They are programs of

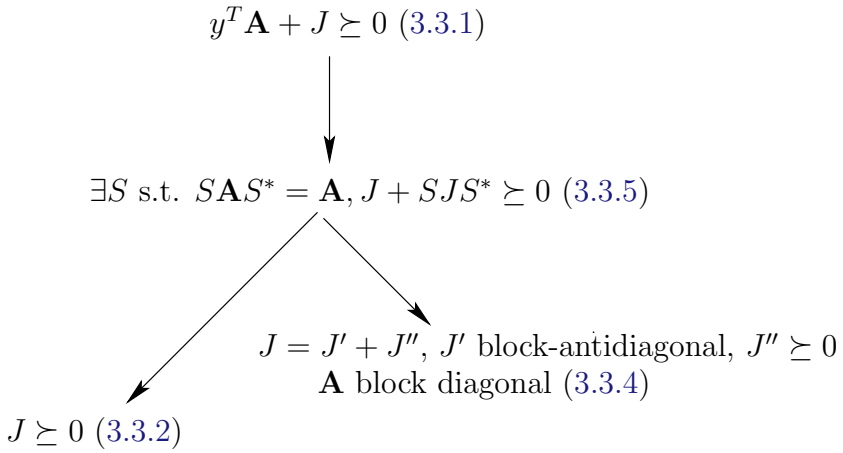

Figure 3.1: Sufficient conditions

the following form:

$$\begin{aligned} \alpha(\pi) = \max_X\ & J \bullet X \\ \text{s.t. } & \mathbf{A} \bullet X = b \\ & \mathbf{B} \bullet X \geq \mathbf{0} \\ & X \succeq 0 \end{aligned}$$

Here both $\mathbf{A}$ and $\mathbf{B}$ are vectors of matrices, and $\mathbf{0}$ denotes the all 0 vector.

Of course, a program of this form can be equivalently written as an affine program by suitably extending X and modifying $\mathbf{A}$ accordingly to enforce the $\mathbf{B} \bullet X \geq \mathbf{0}$ constraints through the $X \succeq 0$ condition. Transforming the format of the program in this way, however, can lead to undesired constraints in the product of instances. The somewhat subtle point is that two equivalent programs do not necessarily lead to equivalent product instances. We explicitly separate out the non-negativity constraints here so that we can define the product as follows: for two programs, $\pi_1 = (J_1, \mathbf{A}_1, b_1, \mathbf{B}_1)$ and $\pi_2 = (J_2, \mathbf{A}_2, b_2, \mathbf{B}_2)$ we say

$$\pi_1 \times \pi_2 = (J_1 \otimes J_2, \mathbf{A}_1 \otimes \mathbf{A}_2, b_1 \otimes b_2, \mathbf{B}_1 \otimes \mathbf{B}_2).$$

Notice that the equality constraints and non-negativity constraints do not interact in the product, which is usually the intended meaning of the product of instances. Indeed, this is the notion of product desired in [FL92, LSŠ08].

It is again straightforward to see that $\alpha(\pi_1 \times \pi_2) \geq \alpha(\pi_1)\alpha(\pi_2)$, thus we focus on the reverse inequality.

Theorem 3.4.1. *Let $\pi_1 = (J_1, \mathbf{A}_1, b_1, \mathbf{B}_1)$ and $\pi_2 = (J_2, \mathbf{A}_2, b_2, \mathbf{B}_2)$ be two semidefinite programs for which strong duality holds. Suppose the following two conditions hold:*

1. *(Bipartiteness) There is a partition of rows and columns into two sets, such that, with respect to this partition, J_i and all the matrices of $\mathbf{B}_i$ are block anti-diagonal, and all the matrices of $\mathbf{A}_i$ are block diagonal, for $i \in \{1, 2\}$.*

2. *There are non-negative vectors u_1, u_2 such that $J_1 = u_1^T \mathbf{B}_1$ and $J_2 = u_2^T \mathbf{B}_2$.*

Then $\alpha(\pi_1 \times \pi_2) \leq \alpha(\pi_1)\alpha(\pi_2)$.

Proof. To prove the theorem it will be useful to consider the dual formulations of π_1 and π_2. Dualizing in the standard fashion, we find

$$\begin{aligned} \alpha(\pi_1) = \min_{y_1, z_1} \; & y_1^T b_1 \\ \text{s.t. } & y_1^T \mathbf{A}_1 - (z_1^T \mathbf{B}_1 + J_1) \succeq 0 \\ & z_1 \geq 0 \end{aligned}$$

and similarly for π_2. Fix y_1, z_1 to be vectors that realize this optimum for π_1 and similarly y_2, z_2 for π_2. The key observation of the proof is that if we can also show that

$$y_1^T \mathbf{A}_1 + (z_1^T \mathbf{B}_1 + J_1) \succeq 0 \text{ and } y_2^T \mathbf{A}_2 + (z_2^T \mathbf{B}_2 + J_2) \succeq 0 \tag{3.4.1}$$

then we will be done. Let us for the moment assume Equation 3.4.1 and see why this is the case.

If Equation 3.4.1 holds, then we also have

$$\begin{aligned} \big(y_1^T \mathbf{A}_1 - (z_1^T \mathbf{B}_1 + J_1)\big) \otimes \big(y_2^T \mathbf{A}_2 + (z_2^T \mathbf{B}_2 + J_2)\big) \succeq 0 \\ \big(y_1^T \mathbf{A}_1 + (z_1^T \mathbf{B}_1 + J_1)\big) \otimes \big(y_2^T \mathbf{A}_2 - (z_2^T \mathbf{B}_2 + J_2)\big) \succeq 0 \end{aligned}$$

Averaging these equations, we find

$$(y_1 \otimes y_2)^T (\mathbf{A}_1 \otimes \mathbf{A}_2) - \big((z_1^T \mathbf{B}_1 + J_1) \otimes (z_2^T \mathbf{B}_2 + J_2)\big) \succeq 0.$$

Let us work on the second term. We have

$$\begin{aligned}(z_1^T\mathbf{B}_1 + J_1) \otimes (z_2^T\mathbf{B}_2 + J_2)\\ = (z_1 \otimes z_2)^T(\mathbf{B}_1 \otimes \mathbf{B}_2) + z_1^T\mathbf{B}_1 \otimes J_2 + J_1 \otimes z_2^T\mathbf{B}_2 + J_1 \otimes J_2\\ = (z_1 \otimes z_2)^T(\mathbf{B}_1 \otimes \mathbf{B}_2) + (z_1 \otimes u_2)^T\mathbf{B}_1 \otimes \mathbf{B}_2 + (u_1 \otimes z_2)^T\mathbf{B}_1 \otimes \mathbf{B}_2 + J_1 \otimes J_2.\end{aligned}$$

Thus if we let $v = z_1 \otimes z_2 + z_1 \otimes u_2 + u_1 \otimes z_2$ we see that $v \geq 0$ as all of z_1, z_2, u_1, u_2 are, and also

$$(y_1 \otimes y_2)^T \otimes (\mathbf{A}_1 \otimes \mathbf{A}_2) - (v^T(\mathbf{B}_1 \otimes \mathbf{B}_2) + J_1 \otimes J_2) \succeq 0.$$

Hence $(y_1 \otimes y_2, v)$ form a feasible solution to the dual formulation of $\pi_1 \times \pi_2$ with value $(y_1 \otimes y_2)(b_1 \otimes b_2) = \alpha(\pi_1)\alpha(\pi_2)$.

It now remains to show that Equation 3.4.1 follows from the condition of the theorem. Given $y\mathbf{A} - (z^T\mathbf{B} + J) \succeq 0$ and the bipartiteness condition of the theorem, we will show that $y\mathbf{A} + (z^T\mathbf{B} + J) \succeq 0$. We have that $y^T\mathbf{A}$ is block diagonal and $z^T\mathbf{B} + J$ is block anti-diagonal with respect to the same partition. So by Lemma 3.3.3, we know $y\mathbf{A} + (z^T\mathbf{B} + J) \succeq 0$. Now apply this argument to both π_1 and π_2. □

One may find the condition that, J lies in the positive span of $\mathbf{B}$, in the statement of Theorem 3.4.1, somewhat unnatural. If we remove this condition, however, a simple counterexample shows that the theorem no longer holds. Consider the program

$$\begin{aligned}\alpha(\pi) = \max_X \begin{bmatrix} 0 & -1 \\ -1 & 0 \end{bmatrix} \bullet X\\ \text{such that } I \bullet X = 1, \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \bullet X \geq 0, X \succeq 0.\end{aligned}$$

Here I stands for the 2-by-2 identity matrix. This program satisfies the bipartiteness condition of Theorem 3.4.1, but J does not lie in the positive span of the matrices of $\mathbf{B}$. It is easy to see that the value of this program is zero. The program $\pi \times \pi$, however, has positive value as $J \otimes J$ does not have any negative entries but is the matrix with ones on the main anti-diagonal.

Note : Theorem 3.4.1 should not be seen as a single theorem. If we look at the proof strategy closely, it can be extended to SDPs which have two sets of constraints. In this case one set was $=$ and other was $\geq$ constraints. We can have any mix of $\geq, \leq, =$ constraints and the corresponding product theorem with required conditions will emerge from the this proof strategy. We will see one such application in the Section 3.4.2.

3.4.1 Discrepancy

Communication complexity is an ideal model to study direct sum and direct product theorems. It is simple enough that one can often hope to attain tight results, yet powerful enough that such theorems are non-trivial and have applications to reasonably powerful models of computation. See [KN97] for more details on communication complexity and its applications. Shaltiel [Sha03] proved a direct product theorem for communication complexity lower bounds shown by a particular method—the discrepancy method under the uniform distribution. Shaltiel does not explicitly use semidefinite programming techniques, but proceeds by relating discrepancy under the uniform distribution to the spectral norm, which can be cast as a semidefinite program.

This result was recently generalized and strengthened by Lee, Shraibman, and Špalek [LSŠ08] who show an essentially optimal direct product theorem for discrepancy under arbitrary distributions. This result follows the general plan for showing product theorems via semidefinite programming: they use a result of Linial and Shraibman [LS08] that a semidefinite programming quantity $\gamma_2^\infty(M)$ characterizes the discrepancy of the communication matrix M up to a constant factor, and then show that $\gamma_2^\infty(M)$ perfectly products. The semidefinite programming formulation of $\gamma_2^\infty(M)$ is not affine but involves non-negativity constraints. Let us now look at the semidefinite program describing γ_2^∞:

$$\begin{aligned}
\gamma_2^\infty(M) = \max_X \ & \widehat{M} \bullet X \text{ such that} \\
& X \bullet I = 1 \\
& X \bullet E_{ij} = 0 \text{ for all } i \neq j \leq m, i \neq j \geq m \\
& X \bullet (\widehat{M} \circ E_{ij}) \geq 0 \text{ for all } i \leq m, j \geq m, \text{ and } i \geq m, j \leq m
\end{aligned}$$

$$X \succeq 0.$$

Here $E_{i,j}$ is the 0/1 matrix with exactly one entry equal to 1 in coordinate (i, j). In this case, $\mathbf{A}$ is formed from the matrices I and E_{ij} for $i \neq j \leq m$ and $i \neq j \geq m$. These matrices are all block diagonal with respect to the natural partition of $\widehat{M}$. Further, the objective matrix $\widehat{M}$ and matrices of $\mathbf{B}$ are all block anti-diagonal with respect to this partition. Finally, we can express $\widehat{M} = u^T\mathbf{B}$ by simply taking u to be the all 1 vector.

3.4.2 Feige-Lovász

In a seminal paper, Babai, Fortnow, and Lund [BFL91] show that all of non-deterministic exponential time can be captured by interactive proof systems with two-provers and polynomially many rounds. The attempt to characterize the power of two-prover systems with just one round sparked interest in a parallel repetition theorem—the question of whether the soundness of a two-prover system can be amplified by running several checks in parallel. Feige and Lovász [FL92] ended up showing that two-prover one-round systems capture NEXP by other means, and a proof of a parallel repetition theorem turned out to be the more difficult question [Raz98]. In the same paper, however, Feige and Lovász also take up the study of parallel repetition theorems and show an early positive result in this direction. In a two-prover one-round game, the Verifier is trying to check if some input x is in the language L. The Verifier chooses questions $s \in S, t \in T$ with some probability $P(s, t)$ and then sends question s to prover Alice, and question t to prover Bob. Alice sends back an answer $u \in U$ and Bob replies $w \in W$, and then the Verifier answers according to some boolean predicate $V(s, t, u, w)$. We call this a game $G(V, P)$, and write the acceptance probability of the Verifier as $\omega(G)$. In much the same spirit as the result of Lovász on the Shannon capacity of a graph, Feige and Lovász show that if the value of a game $\omega(G) < 1$ then also $\sup_k \omega(G^k)^{1/k} < 1$, for a certain class of games known as unique games.

The proof of this result proceeds in the usual way: Feige and Lovász first show that $\omega(G)$ can be represented as a quadratic program. They then relax this quadratic program in the natural way to obtain a semidefinite program with value $\sigma(G) \geq \omega(G)$.

Here the proof faces an extra complication as $\sigma(G)$ does not perfectly product either. Thus another round of relaxation is done, throwing out some constraints to obtain a program with value $\bar{\sigma}(G) \geq \sigma(G)$ which does perfectly product. Part of our motivation for proving Theorem 3.4.1 was to uncover the "magic" of this second round of relaxation, and explain why Feige and Lovász remove the constraints they do in order to obtain something which perfectly products.

Although the parallel repetition theorem was eventually proven by different means [Raz98, Hol07], the semidefinite programming approach has recently seen renewed interest for showing tighter parallel repetition theorems for restricted classes of games and where the provers share entanglement [CSUU07, KRT07, BHH+08].

The relaxed program

As mentioned above, Feige and Lovász first write $\omega(G)$ as an integer program, and then relax this to a semidefinite program with value $\sigma(G) \geq \omega(G)$. We now describe this program. The objective matrix C is a $|S| \times |U|$-by-$|T| \times |W|$ matrix where the rows are labeled by pairs (s, u) of possible question and answer pairs with Alice and similarly the columns are labeled by (t, w) possible dialogue with Bob. The objective matrix for a game $G = (V, P)$ is given by $C[(s, u), (t, w)] = P(s, t)V(s, t, u, w)$. We also define an auxiliary matrices B_{st} of dimensions the same as $\widehat{C}$, where $B_{st}[(s', u), (t', w)] = 1$ if $s = s'$ and $t = t'$ and is zero otherwise.

With these notations in place, we can define the program:

$$\begin{aligned} \sigma(G) = \max_X \frac{1}{2}\widehat{C} \bullet X \text{ such that} & \\ X \bullet B_{st} = 1 \text{ for all } s, t \in S \cup T & \qquad (3.4.2) \\ X \geq 0 & \qquad (3.4.3) \\ X \succeq 0 & \end{aligned}$$

We see that we cannot apply Theorem 3.4.1 here as we have global non-negativity constraints (not confined to the off-diagonal blocks) and global equality constraints (not confined to the diagonal blocks). Indeed, Feige and Lovász remark that this program

does not perfectly product. Also Kempe and Regev ([KR10]) showed that a similar SDP (slightly tighter with additional orthogonality constraints) does not product.

Feige and Lovász then consider a further relaxation with value $\bar{\sigma}(G)$ whose program does fit into our framework. They throw out all the constraints of Equation 3.4.2 which are off-diagonal, and remove the non-negativity constraints for the on-diagonal blocks of X, Equation 3.4.3. More precisely, they consider the following program:

$$\begin{aligned}
\bar{\sigma}(G) = \max_X \ & \frac{1}{2}\widehat{C} \bullet X \text{ such that} \\
& \sum_{u,w \in U} |X[(s,u),(s',w)]| \leq 1 \text{ for all } s, s' \in S \qquad (3.4.4) \\
& \sum_{u,w \in W} |X[(t,u),(t',w)]| \leq 1 \text{ for all } t, t' \in T \qquad (3.4.5) \\
& X \bullet E_{(s,u),(t,w)} \geq 0 \text{ for all } s \in S, t \in T, u \in U, w \in W \\
& X \succeq 0
\end{aligned}$$

Let us see that this program fits into the framework of Theorem 3.4.1. The vector of matrices $\mathbf{B}$ is composed of the matrices $E_{(s,u),(t,w)}$ for $s \in S, u \in U$ and $t \in T, w \in W$. Each of these matrices is block diagonal with respect to the natural partition of $\widehat{C}$. Moreover, as $\widehat{C}$ is non-negative and bipartite, we can write $\widehat{C} = u^T \mathbf{B}$ for a non-negative u, namely where u is given by concatenation of the entries of C and C^T written as a long vector.

The on-diagonal constraints given by Equations 3.4.4 and Eq. 3.4.5 are not immediately seen to be of the form needed for Theorem 3.4.1 for two reasons: first, they are inequalities rather than equalities, and second, they have of absolute value signs. Fortunately, both of these problems can be easily dealt with.

It is not hard to check that Theorem 3.4.1 also works for inequality constraints $\mathbf{A} \bullet X \leq b$. The only change needed is that in the dual formulation we have the additional constraint $y \geq \mathbf{0}$. This condition is preserved in the product solution constructed in the proof of Theorem 3.4.1 as $y \otimes y \geq 0$.

The difficulty in allowing constraints of the form $\mathbf{A} \bullet X \leq b$ is in fact that the opposite direction $\alpha(\pi_1 \times \pi_2) \geq \alpha(\pi_1)\alpha(\pi_2)$ does not hold in general. Essentially, what can go wrong here is that $a_1, a_2 \leq b$ does not imply $a_1 a_2 \leq b^2$. In our case,

however, this does not occur as all the terms involved are positive and so one can show $\bar{\sigma}(G_1 \times G_2) \geq \bar{\sigma}(G_1)\bar{\sigma}(G_2)$.

To handle the absolute value signs we consider an equivalent formulation of $\bar{\sigma}(G)$. We replace the condition that the sum of absolute values is at most one by constraints saying that the sum of every possible $\pm$ combination of values is at most one:

$$\begin{aligned}
\bar{\sigma}'(G) = &\max_X \frac{1}{2}\widehat{C} \bullet X \text{ such that} \\
&\sum_{u,w \in U} (-1)^{x_{uw}} X[(s,u),(s',w)] \leq 1 \text{ for all } s, s' \in S \text{ and } x \in \{0,1\}^{|U|^2} \\
&\sum_{u,w \in W} (-1)^{x_{uw}} X[(t,u),(t',w)] \leq 1 \text{ for all } t, t' \in T \text{ and } x \in \{0,1\}^{|W|^2} \\
&X \bullet E_{(s,u),(t,w)} \geq 0 \text{ for all } s \in S, t \in T, u \in U, w \in W \\
&X \succeq 0
\end{aligned}$$

This program now satisfies the conditions of Theorem 3.4.1. It is clear that $\bar{\sigma}(G) = \bar{\sigma}'(G)$, and also that this equivalence is preserved under product. Thus the product theorem for $\bar{\sigma}(G)$ follows from Theorem 3.4.1 as well.

3.5 Extension to linear programming

It is known that linear programming is a special case of semidefinite programming. So it is natural to ask what these conditions tell us about product rules in linear programming.

Consider a standard linear program $\pi(c, A, b)$:

$$\begin{aligned}
\max_x \quad & c^T x \\
s.t.\ \forall i \in [m] \quad & a_i^T x = b_i \\
& x \geq 0
\end{aligned}$$

Where a_i are the rows of matrix A, so the middle constraint is $Ax = b$.

Let C be the diagonal matrix, whose diagonal is c. Similarly for every i, define the diagonal matrix A_i, with a_i as its diagonal. Then the above linear program can be

written as the SDP

$$\begin{aligned} \max_{X} \quad & C \bullet X \\ s.t.\ \forall i \in [m] \quad & A_i \bullet X = b_i \\ & X \succeq 0 \end{aligned}$$

If there is a solution x to the linear program, then the diagonal matrix X with x as the diagonal, is a solution of the above SDP with the same value. Similarly, if there is a solution matrix X of SDP, its diagonal x will be a solution of the LP with the same objective value. So these two programs are equivalent.

The condition discussed in Section 3.3.1, gives a product rule for linear programming. The matrix C is positive semidefinite if and only if all the coefficients in the objective vector c are positive. So we get the theorem,

Theorem 3.5.1. *Let $\pi_1(c_1, A_1, b_1)$ and $\pi_2(c_2, A_2, b_2)$ be two standard instances of linear programming. If both objective vectors are entry wise positive, i.e., $c_1, c_2 \geq 0$, then*

$$\alpha(\pi_1 \otimes \pi_2) = \alpha(\pi_1)\alpha(\pi_2),$$

where α denotes the optimum of the linear program.

The condition discussed in Section 3.3.2 cannot be extended to linear programming. Since when you convert an LP into an SDP, the resulting matrices will be diagonal matrices.

3.6 A necessary condition for dual feasibility

In this section, we turn our attention to necessary conditions for a product theorem to hold. If we restrict our attention to an instance and its square, we are able to give a condition which is necessary and sufficient, but is somewhat unsatisfactory as the expression explicitly refers to the optimal dual solutions y_1 and y_2, like dual feasibility itself. It remains a task for the future to develop a necessary and sufficient condition whose criterion is formulated solely in terms of the problem instances π_1 and π_2.

Assume y_1, y_2 to be the optimal dual solutions for π_1, π_2. We know that $y_1^T\mathbf{A}_1 - J_1, y_2^T\mathbf{A}_2 - J_2$ are positive semidefinite. Consider the trivial case when both of them are 0. Barring this trivial case when both of them are zero, we can show that sufficient condition given in Lemma 3.3.1 ($y_c^T\mathbf{A}_c + J_c \succeq 0$) is partially necessary. More formally

Theorem 3.6.1. *For two semidefinite programming instances π_1 and π_2, let y_1 and y_2 be optimal solutions of π_1^* and π_2^*, respectively. Further, assume that at least one of $y_c^T\mathbf{A}_c - J_c$ is not identically zero for $c \in \{1, 2\}$. Then $y_1 \otimes y_2$ is a feasible solution of the dual of the product instance (i.e., Condition 2 of Section 3.2.2 holds) only if at least one of $y_c^T\mathbf{A}_c + J_c \succeq 0$ for $c \in \{1, 2\}$.*

Proof. We will assume that $y_c^T\mathbf{A}_c + J_c \not\succeq 0$ for both $c \in \{1, 2\}$ and show that this implies that $y_1 \otimes y_2$ is not dual feasible for $\pi_1 \times \pi_2$. Lets divide the proof into 3 cases depending upon whether any $y_c^T\mathbf{A}_c - J_c, c \in \{0, 1\}$ is strictly zero.

Case 1: $y_1^T\mathbf{A}_1 - J_1 = y_2^T\mathbf{A}_2 - J_2 = 0$.

By our assumption in the theorem, this case can't happen. But for the sake of completeness, in this case $(y_1 \otimes y_2)^T(\mathbf{A}_1 \otimes \mathbf{A}_2) - J_1 \otimes J_2 = 0$, so $y_1 \otimes y_2$ is dual feasible for $\pi_1^* \times \pi_2^*$ and the product theorem holds. Note that it is not necessary for one of $y_1^T\mathbf{A}_1 + J_1, y_2^T\mathbf{A}_2 + J_2$ to be positive semidefinite.

Case 2: Exactly one of $y_c^T\mathbf{A}_c - J_c$ is zero (say $y_1^T\mathbf{A}_1 - J_1$).

Then $(y_1^T\mathbf{A}_1 - J_1) \otimes (y_2^T\mathbf{A}_2 + J_2)$ is zero. Hence

$$2(y_1^T\mathbf{A}_1 \otimes y_2^T\mathbf{A}_2 - J_1 \otimes J_2) = (y_1^T\mathbf{A}_1 + J_1) \otimes (y_2^T\mathbf{A}_2 - J_2), \tag{3.6.1}$$

and $y_1 \otimes y_2$ is dual feasible if and only if $y_1^T\mathbf{A}_1 + J_1$ is positive semidefinite.

Case 3: None of $y_c^T\mathbf{A}_c - J_c$ is zero.

By our assumption that $y_c^T\mathbf{A}_c + J_c \not\succeq 0$, we have vectors v_c, such that $v_c^T(y_c^T\mathbf{A}_c + J_c)v_c < 0$ for both $c \in \{1, 2\}$. As y_1, y_2 are dual feasible, we also know that $v_c^T(y_c^T\mathbf{A}_c - J_c)v_c \geq 0$ for both $c \in \{1, 2\}$. We would actually like this second inequality to be strict.

Claim 3.6.2. *There exist vectors w_c, s.t.,*

$$w_c^T(y_c^T\mathbf{A}_c + J_c)w_c < 0, \quad w_c^T(y_c^T\mathbf{A}_c - J_c)w_c > 0$$

Proof. Suppose $w_c = v_c$ does not salsify these conditions. First observe that

$$\begin{aligned} & y_c^T \mathbf{A}_c - J_c \succeq 0 \\ \Rightarrow \quad & \exists B \quad \text{s.t.} \quad y_c^T \mathbf{A}_c - J_c = BB^T \\ \Rightarrow \quad & Bv = 0 \ \ (\text{Since } (y_c^T \mathbf{A}_c - J_c)v_c = 0 \ ) \\ \Rightarrow \quad & (y_c^T \mathbf{A}_c - J_c)v_c = 0 \ \ \ c \in \{1,2\} \end{aligned} \tag{3.6.2}$$

Secondly, we know that $y_c^T \mathbf{A}_c - J_c \succeq 0$ and not the all zero matrix. Then there exist vectors u_c for both $c \in \{1,2\}$, such that $u_c^T (y_c^T \mathbf{A}_c - J_c)u_c > 0$.

Now, let $w_c = v_c + \delta_c u_c$, for some parameter δ_c. From Equation 3.6.2 and appropriately small δ, we obtain $w_c^T (y_c^T \mathbf{A}_c + J_c)w_c < 0$ and $w_c^T (y_c^T \mathbf{A}_c - J_c)w_c > 0$ for $c = 1, 2$. □

With these new w_c's from Claim 3.6.2, construct $w = w_1 \otimes w_2$. Then

$$\begin{aligned} 0 \ &> \ w^T((y_1^T \mathbf{A}_1 - J_1) \otimes (y_2^T \mathbf{A}_2 + J_2))w \\ &= \ w^T(y_1^T \mathbf{A}_1 \otimes y_2^T \mathbf{A}_2 - J_1 \otimes J_2 - J_1 \otimes y_2^T \mathbf{A}_2 + y_1^T \mathbf{A}_1 \otimes J_2)w \\ &= \ w^T(y_1^T \mathbf{A}_1 \otimes y_2^T \mathbf{A}_2 - J_1 \otimes J_2)w + w^T(y_1^T \mathbf{A}_1 \otimes J_2 - J_1 \otimes y_2^T \mathbf{A}_2)w \end{aligned}$$

By similar argument, considering now the inequality

$$w^T((y_1^T \mathbf{A}_1 + J_1) \otimes (y_2^T \mathbf{A}_2 - J_2))w < 0,$$

we can show that

$$w^T(y_1^T \mathbf{A}_1 \otimes y_2^T \mathbf{A}_2 - J_1 \otimes J_2)w + w^T(-y_1^T \mathbf{A}_1 \otimes J_2 + J_1 \otimes y_2^T \mathbf{A}_2)w \ \ < \ \ 0$$

By averaging the two inequalities we get that

$$w^T(y_1^T \mathbf{A}_1 \otimes y_2^T \mathbf{A}_2 - J_1 \otimes J_2)w < 0$$

which implies that $y_1 \otimes y_2$ is not feasible for $\pi_1 \times \pi_2$. □

One might suspect that the full converse of Lemma 3.3.1 holds, i.e., in the case of the feasibility of $y_1 \otimes y_2$, both $y_1^T \mathbf{A}_1 + J_1$ and $y_2^T \mathbf{A}_2 + J_2$ should be positive semidefinite. We now show that this is not necessarily the case.

Lets return to the example of Section 3.2.1. Recall that in this example, $J = M$, $\mathbf{A} = (I)$ and $y = \lambda$ (the maximal eigenvalue of M). Let M_1 be a symmetric matrix with eigenvalues -2 and 1 and let M_2 be a symmetric matrix with eigenvalues 0 and 1. Then $y_1 = 1$ and $y_2 = 1$, so $y_1 \otimes y_2 = 1$, which is a solution of

$$\{\min \lambda \mid \lambda I - M_1 \otimes M_2 \succeq 0\}, \tag{3.6.3}$$

even though $I + M_1$ is not positive semidefinite.

3.7 The weak product

A surprising observation about the theta number of Lovász, well described in [Knu94], is that it is multiplicative with two different notions of products:

Definition 3.7.1 (Strong product "$\times$" of graphs)**.** *$(u', u'') -- (v', v'')$ or $(u', u'') = (v', v'')$ in $G' \times G''$ if and only if ($u' -- v'$ or $u' = v'$ in G') and ($u'' -- v''$ or $u'' = v''$ in G'').*

and

Definition 3.7.2 (Weak product "$\times_w$" of graphs)**.** $G' \times_w G'' = \overline{\overline{G'} \times \overline{G''}}$.

Recall that $\vartheta(G)$ is defined by [Sze94] (by J we denote the matrix with all 1 elements):

$$\vartheta(G) = \max\{J \bullet X \mid \ I \bullet X = 1;\ \ \forall (i,j) \in E(G) :\ \ X_{i,j} = 0;\ X \succeq 0\}. \tag{3.7.1}$$

That is, every edge gives a new linear constraint, increasing m by one. In general, $E(G' \times_w G'') \supseteq E(G' \times G'')$, because $(u', u'') -- (v', v'')$ is an edge of $G' \times G''$ if and only if both of its projections are edges or identical coordinates, but $(u', u'') \neq (v', v'')$. On the other hand, $(u', u'') -- (v', v'')$ is an edge of $G' \times_w G''$ if and only if there exists at least one projection which is an edge.

It is easy to see that the constraint in Expression 3.7.1 for $\vartheta(G' \times G'')$ has a constraint for every constraint pair in the corresponding expression for G' and G''. So the strong product is the one that corresponds to our usual product notion that appears in previous

sections. In contrast, when we write down Expression 3.7.1 for $\vartheta(G' \times_w G'')$, we see a lot of extra constraints.

How do they arise? In general, assume that we know that the product solution $X_1 \otimes X_2$ is the optimal solution for $\pi_1 \times \pi_2$ (which is indeed the case under the conditions we considered in earlier sections). Assume furthermore that some coordinate i of b_1 is zero. Then $A_1^{(i)} \bullet X_1 = 0$. Now we may take any $n_2 \times n_2$ matrix B, and it will hold that

$$(A_1^{(i)} \otimes B) \bullet (X_1 \otimes X_2) = (A_1^{(i)} \bullet X_1)(B \bullet X_2) = 0.$$

Therefore adding matrices of the form $A_1^{(i)} \otimes B$ to $\mathbf{A}_1 \otimes \mathbf{A}_2$ and setting the the corresponding entry of the longer b vector of the product instance to zero will not influence the objective value. The same can be said when the roles of π_1 and π_2 are exchanged.

We can easily see that the weak product in the case of the theta number arises this way. What equations we wish to add to the product system this way is a matter of taste, and we believe it depends on the specific class of semidefinite programming instances under study. We summarize the finding of this section in the following proposition

Proposition 3.7.3. *Assume that for affine instances π_1 and π_2 the product rule holds. Then if we define a system $\pi_1 \times_w \pi_2$ that we call "weak product" by conveniently adding an arbitrary number of new constrains to the system that follow the construction rules described above (in particular, every added constraint should be associated with a zero entry of b_1 or b_2), the product rule will also hold for the weak product.*

The above lemma explains why the theta number of Lovász obeys the product rule with respect to the weak product of graphs.

3.8 Discussion

We have begun to systematically investigate product theorems for affine instances of semidefinite programming. Our theorems are able to explain many of the semidefinite program product theorems in the literature, including [Lov79, FL92, CSUU07, KRT07, LSŠ08]. This list of examples allows us to conclude that we have hit upon a basic research topic with immediate and multiple applications in computer science. Indeed,

we hope that the framework we have developed will encourage further applications of the method of proving product theorems via semidefinite programming.

While the sufficient conditions given here are able to explain many of the examples in the literature, we still feel that such a natural problem of when semidefinite programs obey a product rule deserves further research to obtain a more elegant and cohesive mathematical theory.

Another direction of research is to try for more general *composition* theorems rather than just product theorems. This is motivated by the common setting in boolean function complexity where, say, one has a lower bound on the complexity of $f : \{0,1\}^n \to \{0,1\}$ and $g : \{0,1\}^k \to \{0,1\}$, and would like to obtain a lower bound on $(f \circ g)(\vec{x}) = f(g(x_1), \ldots, g(x_n))$. What we have studied so far in looking at tensor products corresponds to the special cases where f is the PARITY or the AND function, depending on whether the objective matrix is a sign matrix or a 0/1 valued matrix. One example of such a general composition theorem is known for the adversary method, a semidefinite programming quantity which lower bounds quantum query complexity. There it holds that $\mathrm{Adv}(f \circ g) \geq \mathrm{Adv}(f)\mathrm{Adv}(g)$ [Amb03, HLŠ07]. It would be interesting to develop a theory to explain these cases as well.

Chapter 4

Quantum query complexity

Quantum query complexity measures the number of coherent, black-box queries to the input string needed to evaluate a function. Many quantum algorithms can be formulated in the query model, and the model has the further advantage that strong lower bounds can often be shown.

One of the main techniques for placing lower bounds on quantum query complexity is the adversary method. The origins of the adversary bound can be traced to the hybrid argument of Bennett et al.[BBBV97]. Ambainis developed the adversary method proper[Amb02] and subsequently many alternative formulations were given[HNS02, BS04, Amb06, Zha05, BSS03, LM04]—all later shown to be equivalent[ŠS06]. Finally, the bound was modified to allow negative weights, resulting in a strictly stronger bound known as the general adversary bound[HLŠ07]. The general adversary bound of a function f, which we will denote as $\mathrm{Adv}^{\pm}(f)$, can be written as a semi-definite program (SDP), and behaves well under function composition.

A recent sequence of works [FGG08, CCJY09, ACR+10, RŠ08] has culminated in showing that the general adversary bound characterizes the bounded-error quantum query complexity of a boolean function, up to a constant factor[Rei10a, Rei10b, Rei10c]. Our work completes this picture by showing that, up to a constant factor, the general adversary method characterizes the bounded-error quantum query complexity of any function whatsoever, boolean or non-boolean, partial or total.

Theorem 4.0.1. *For finite sets C and E, and $\mathcal{D} \subseteq C^n$, let $f : \mathcal{D} \to E$. Then*

$$Q(f) = \Theta\big(\mathrm{Adv}^{\pm}(f)\big) \ . \tag{4.0.1}$$

Compared to the boolean case, the non-boolean case presents new challenges. What

emerges from our work is that for *upper bounds* it is key to use a slightly different SDP than the adversary bound. When phrased as a minimization problem, this new program, which we call witness size, can be viewed as the general adversary bound with additional constraints. While the adversary bound only has constraints on input pairs x, y with $f(x) \neq f(y)$, the witness size has constraints on all pairs including those where $f(x) = f(y)$. These additional constraints are crucial for the construction of our algorithm.

From this description it is clear that the witness size value will be at least as large as the general adversary bound. We show the lucky fact that it can be at most a factor of two larger. For a boolean function, the witness size SDP value equals the least *span program* witness size, a complexity measure on boolean functions that has been shown to equal the general adversary bound [Rei09, Rei10b].

Our algorithm essentially runs a quantum walk on a certain weighted bipartite graph constructed from a solution to the witness size SDP. Figure 4.1 shows an example. Compared to the graphs used for evaluating boolean functions, there are two main new features. To allow for a non-boolean input alphabet, we add certain weighted star graphs (dashed) depending on the input letters instead of single dangling edges. To accommodate non-boolean output, we have multiple output vertices instead of just one. The edges are weighted according to the witness size SDP solution instead of according to the general adversary bound SDP.

In our second result, we show that quantum query complexity possesses a remarkable algorithmic property, $Q\big(f(g(x^1), \ldots, g(x^n))\big) = O\big(Q(f)Q(g)\big)$ for any compatible functions f, g. This was previously only known in the boolean case. This result follows by showing that the witness size obeys such a composition rule. Again the extra constraints in the witness size program prove crucial in this proof. Extending a result in[HLŠ07], we also show that $Q\big(f(g(x^1), \ldots, g(x^n))\big) = \Omega\big(Q(f)Q(g)\big)$ whenever the output of g, and therefore input of f, is boolean.

Many well-studied functions in quantum query complexity, like element distinctness [BDH+00, Shi02, Amb07, CE05, MSS05] and the ordered search problem [FGGS99, HNS02, CLP07, BOH07, CL08], have non-boolean input or output. It is our hope that

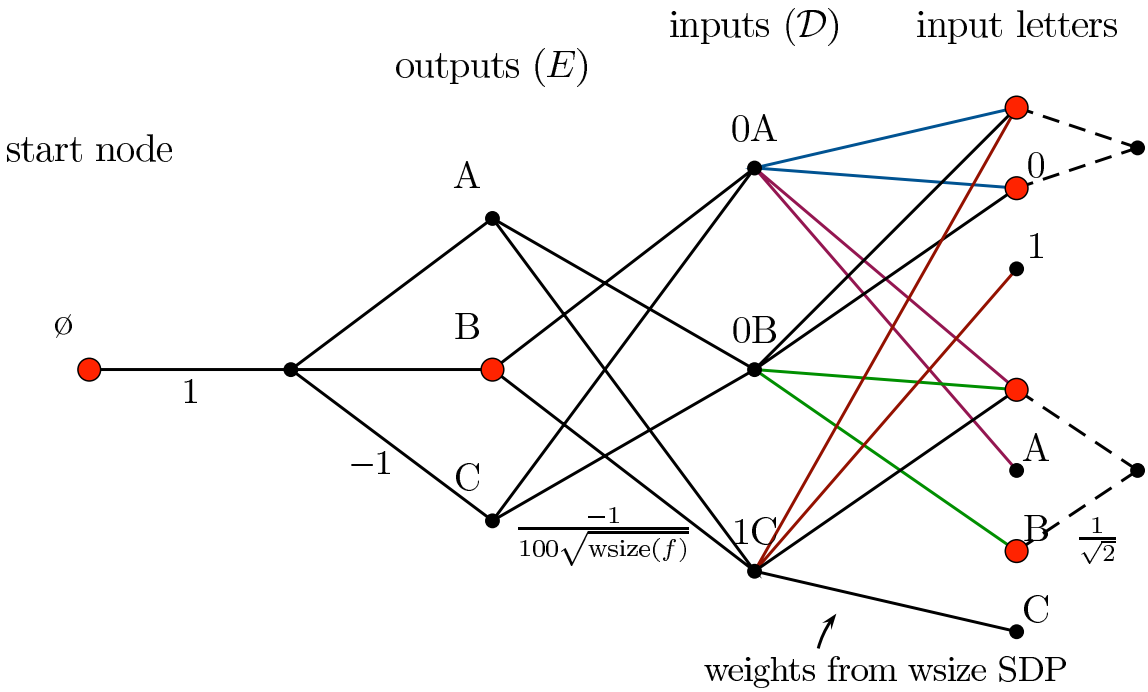

Figure 4.1: An example graph $G(x)$ for a function $f(x_1x_2) = x_2$ mapping $\mathcal{D} = \{0A, 0B, 1C\} \subset \{0,1\} \times \{\text{A}, \text{B}, \text{C}\}$ to $E = \{\text{A}, \text{B}, \text{C}\}$. Our algorithm essentially runs a quantum walk on the vertices of this bipartite graph, starting at $|\text{ø}\rangle$. The walk converges, on average, to an eigenvalue-zero eigenvector supported on the larger, red vertices, after which measuring the vertex has a good probability of giving output vertex $f(0\text{B}) = \text{B}$. (In general, there will be many more vertices in the last two levels from ø.) The dashed lines are added to the graph G to define $G(x)$, shown for input $x = 12$. The details of this construction are given in Section 4.4.

our characterization can help with problems like finding an explicit optimal adversary lower bound for element distinctness, showing that the quantum complexity of ordered search is $(1/\pi)\ln(n)$, or deriving new algorithms for other non-boolean functions.

4.1 Definitions

For a natural number $n \in \mathbf{N}$, let $[n] = \{1, 2, \ldots, n\}$. As discussed before, for two matrices A, B of the same size, $A \circ B$ denotes their entry wise product, also known as Hadamard or Schur product.

For a finite set X, let $\mathbf{C}^X$ be the Hilbert space $\mathbf{C}^{|X|}$ with orthonormal basis $\{|x\rangle : x \in X\}$. For vector spaces V and W over $\mathbf{C}$, let $\mathcal{L}(V,W)$ denote the set of all linear transformations from V into W, and let $\mathcal{L}(V) = \mathcal{L}(V,V)$. $\|A\|$ is the spectral norm of an operator A and $\|A\|_{\mathrm{tr}}$ is the trace norm of A, that is the sum of the singular values of A.

A weighted bipartite graph G can be specified by its weighted biadjacency matrix B_G. G has a vertex for every row and for every column of B_G, and edges between the row and column vertices have weights specified by the matrix entries. The weighted adjacency matrix of G is

$$A_G = \begin{pmatrix} 0 & B_G \\ B_G^\dagger & 0 \end{pmatrix} \quad . \tag{4.1.1}$$

4.1.1 Quantum query complexity

As with the classical model of decision trees, in the quantum query model, we wish to compute some function f and access the input through queries. The complexity of f is the number of queries needed to compute f on a worst-case input x. Unlike the classical case, however, quantum queries can be made in superposition.

For $C = \{0, 1, \ldots, k-1\}$, a finite set E, and $\mathcal{D} \subseteq C^n$, consider a function $f : \mathcal{D} \to E$. The state of a quantum query algorithm for f resides in a space $H_Q \otimes H_W$ where $H_Q = \mathbf{C}^{\{0,\ldots,n\}} \otimes \mathbf{C}^C$ is a $(n+1)k$-dimensional query register with basis $|j,b\rangle$ for $0 \le j \le n$, $0 \le b \le k-1$ and H_W is a workspace of arbitrary dimension. The query operator for x, denoted O_x is defined by the action

$$O_x|j,b\rangle|w\rangle = |j, b + x_j \bmod k\rangle|w\rangle \quad , \tag{4.1.2}$$

where $x_0 = 0$ by convention. On input x, the algorithm begins in the state $|\psi_x^0\rangle = |0\rangle|0\rangle$ and alternately applies a unitary operation independent of the input x and either the query operator O_x or its inverse. The output is determined by a complete orthogonal set of projectors $\{\Pi_b\}_{b \in E}$ labeled by the possible outputs of f. The probability the algorithm outputs b on input x is $\|\Pi_b|\psi_x^T\rangle\|^2$, where $|\psi_x^T\rangle$ is the state of the algorithm on input x after T queries. The 1/3-error quantum query complexity of a function f,

denoted $Q(f)$, is the minimum number of queries made by an algorithm which outputs $f(x)$ with probability at least $2/3$ for every x.

4.1.2 Hadamard product operator norm

We will make use of the γ_2 norm, also known as the Hadamard product operator norm[Bha07]. This norm has been introduced recently to complexity theory by Linial et al.[LMSS07]. It is defined for a matrix A as

$$\gamma_2(A) = \min_{X,Y:\, XY=A} r(X)\, c(Y) \ , \tag{4.1.3}$$

where $r(X)$ is the largest ℓ_2 norm of a row of X and similarly $c(Y)$ is the largest ℓ_2 norm of a column of Y. Using this definition it is straightforward to see that γ_2 is a norm. By writing the optimization problem (4.1.3) as an SDP and taking the dual, an alternative formulation can be derived[LSŠ08]:

$$\gamma_2(A) = \max_{X:\|X\|_{\mathrm{tr}}=1} \|A \circ X\|_{\mathrm{tr}} \ . \tag{4.1.4}$$

The following straightforward lemma plays a key role in relating the adversary bound and witness size, and in the design of our algorithm.

Lemma 4.1.1. *Let S be a finite set and $S_1, \ldots, S_k$ a partition of S. Consider the $|S|$-by-$|S|$ matrix M with entries $M_{i,j} = 0$ if $i, j \in S_b$ for some b, and $M_{i,j} = 1$ otherwise. Then $\gamma_2(M) = 2\big(1 - \frac{1}{k}\big)$.*

Proof. From the formulation of γ_2 given by Eq. (4.1.3) it is readily seen that γ_2 is invariant under adding or removing duplicate rows or columns. By removing duplicate rows and columns, it suffices to consider the k-by-k matrix C with ones everywhere except for zeros on the diagonal.

It is easy to see that $\gamma_2(C) \leq 2$ as C is the difference of the all-ones matrix and the identity matrix, both of which have γ_2 value one. This observation can also be used to give an explicit factorization of C with value 2 by defining $(k+1)$-dimensional vectors $|\mu_i\rangle = |0\rangle + |i\rangle$ and $|\nu_i\rangle = |0\rangle - |i\rangle$ for $i \in [k]$. Then $C_{ij} = \langle \mu_i | \nu_j \rangle$, and each $|\mu_i\rangle, |\nu_i\rangle$ has norm $\sqrt{2}$.

One can actually do slightly better than this. We exhibit unit vectors $|\mu_i\rangle, |\nu_i\rangle \in \mathbf{C}^{[k]}$ for $i \in [k]$ satisfying $\langle \mu_i | \nu_j \rangle = \frac{1}{2}\frac{k}{k-1} C_{ij}$. Let $\alpha = \sqrt{\frac{1}{2} - \frac{\sqrt{k-1}}{k}}$. The vectors are defined as

$$|\mu_i\rangle = -\alpha |i\rangle + \frac{\sqrt{1-\alpha^2}}{\sqrt{k-1}} \sum_{j \neq i} |j\rangle \qquad |\nu_i\rangle = \sqrt{1-\alpha^2}|i\rangle + \frac{\alpha}{\sqrt{k-1}} \sum_{j \neq i} |j\rangle \ . \tag{4.1.5}$$

For the lower bound we use the formulation given in Eq. (4.1.4). The trace norm of C is $2(k-1)$. Taking X to be the matrix with all entries equal to $1/k$ gives the lower bound. □

4.2 The general adversary bound and witness size SDPs

In this section, we will define two semi-definite programs, the general adversary bound and the witness size. The former SDP is from[HLŠ07] and the latter SDP naturally extends a definition in[RŠ08, Rei10b]. We then derive relationships between the SDPs.

For finite sets D and E, and $\mathcal{D} \subseteq D^n$, consider a function $f : \mathcal{D} \to E$. Let $s \in [0, \infty)^n$ be a vector of "costs." Let F and Δ_j, for $j \in [n]$, be the matrices in $\mathcal{L}(\mathbf{C}^{\mathcal{D}})$ defined by

$$F = \sum_{x,y \in \mathcal{D} :\, f(x) \neq f(y)} |x\rangle\langle y| \qquad \Delta_j = \sum_{x,y \in \mathcal{D} :\, x_j \neq y_j} |x\rangle\langle y| \ . \tag{4.2.1}$$

Thus the x, y entry of F, $\langle x|F|y\rangle$, is 1 if and only if $f(x) \neq f(y)$, while the x, y entry of Δ_j is 1 if and only if $x_j \neq y_j$.

Definition 4.2.1. *The general adversary bound for f, with costs s, is denoted* $\mathrm{Adv}^{\pm}_s(f)$*, and is the common optimum value of the primal and dual SDPs:*

$$\max \Big\{ \|\Gamma \circ F\| : \forall j \in [n], \|\Gamma \circ \Delta_j \circ F\| \leq s_j \Big\} \tag{4.2.2}$$

$$= \min \Big\{ \max_{x \in \mathcal{D}} \sum_{j \in [n]} s_j \langle x|(X_j + Y_j)|x\rangle : \sum_{j \in [n]} (X_j - Y_j) \circ \Delta_j \circ F = F \Big\} \ . \tag{4.2.3}$$

The witness size for f, with costs s, is denoted $\mathrm{wsize}_s(f)$*, and is the common optimum value of the primal and dual SDPs:*

$$\max \Big\{ \|\Gamma \circ F\| : \forall j \in [n], \|\Gamma \circ \Delta_j\| \leq s_j \Big\} \tag{4.2.4}$$

$$= \min \Big\{ \max_{x \in \mathcal{D}} \sum_{j \in [n]} s_j \langle x|(X_j + Y_j)|x\rangle : \sum_{j \in [n]} (X_j - Y_j) \circ \Delta_j = F \Big\} \ . \tag{4.2.5}$$

In both cases, the maximization is over real, symmetric matrices $\Gamma \in \mathcal{L}(\mathbf{R}^{\mathcal{D}})$ *and the minimization is over positive semi-definite real matrices* $X_j, Y_j \succeq 0$*, for* $j \in [n]$*. When the cost vector* s *is not specified, it is taken to be* $\vec{1} = (1, 1, \ldots, 1)$*.*

Observe that in the general adversary bound primal SDP Eq. (4.2.2), Γ appears only as $\Gamma \circ F$. Therefore we may restrict $\Gamma = \Gamma \circ F$ without loss of generality. A matrix Γ with $\Gamma = \Gamma \circ F$ is known as an adversary matrix for f.

We will mostly work with the dual of the witness size SDP. Notice that from the perspective of the dual, a witness size solution satisfies more constraints. Namely, in the general adversary bound the constraints only involve terms $\langle x|(X_j - Y_j)|y\rangle$ when $f(x) \neq f(y)$, whereas in the witness size dual we also have constraints on $\langle x|(X_j - Y_j)|y\rangle$ when $f(x) = f(y)$. It is this extra property that is key for showing a composition theorem for witness size and constructing an algorithm for functions with non-boolean output.

In the case of boolean input it is known for the general adversary bound that the variables Y_j can always be eliminated without cost by suitably modifying the X_j, see [Rei10b, Theorem 4.4]; the same is true for the witness size program.

For presenting our algorithm and composition results, it will be useful to work with vector-factorized presentations of the witness size dual SDP in Eq. (4.2.5). Let us define two vector optimization problems, each equivalent to the witness size.

Lemma 4.2.2. *Let* $A_s(F)$ *and* $B_s(F)$ *be defined by*

$$A_s(F) = \min_{\substack{m \in \mathbf{N}, \\ |\mu_{xj}\rangle, |\nu_{xj}\rangle \in \mathbf{C}^m}} \max_{x \in \mathcal{D}} \sum_{j \in [n]} s_j \left(\||\mu_{xj}\rangle\|^2 + \||\nu_{xj}\rangle\|^2 \right) \tag{4.2.6}$$

$$s.t.\ \forall x, y \in \mathcal{D}, \sum_{j \in [n] : x_j \neq y_j} \left(\langle \mu_{xj} | \mu_{yj} \rangle - \langle \nu_{xj} | \nu_{yj} \rangle \right) = \langle x | F | y \rangle$$

$$B_s(F) = \min_{\substack{m \in \mathbf{N}, \\ |u_{xj}\rangle, |v_{xj}\rangle \in \mathbf{C}^m}} \max_{x \in \mathcal{D}} \max \left\{ \sum_{j \in [n]} s_j \||u_{xj}\rangle\|^2, \sum_{j \in [n]} s_j \||v_{xj}\rangle\|^2 \right\} \tag{4.2.7}$$

$$s.t.\ \forall x, y \in \mathcal{D}, \sum_{j \in [n] : x_j \neq y_j} \langle u_{xj} | v_{yj} \rangle = \langle x | F | y \rangle$$

Then

$$\mathrm{wsize}_s(f) = A_s(F) = B_s(F) \ . \tag{4.2.8}$$

The corresponding optimization problems with constraints only on pairs $x, y \in \mathcal{D}$ with $f(x) \neq f(y)$ have optimal values equal to $\mathrm{Adv}^{\pm}_s(f)$.

Proof. The fact that $\mathrm{wsize}_s(f) = A_s(F)$ follows from the straightforward factorization of the positive semi-definite matrices X_j, Y_j in the dual form of witness size, Eq. (4.2.5). To see the equivalence, let $\langle x|X_j|y\rangle = \langle \mu_{xj}|\mu_{yj}\rangle$ and $\langle x|Y_j|y\rangle = \langle \nu_{xj}|\nu_{yj}\rangle$.

To show that $A_s(F) = B_s(F)$, observe that we may assume the vectors in an optimal solution to $A_s(F)$ satisfy $\langle \mu_{xi}|\nu_{yj}\rangle = 0$ for all $x, y \in \mathcal{D}$, $i, j \in [n]$. This means that $\||\mu_{xj}\rangle + |\nu_{xj}\rangle\|^2 = \||\mu_{xj}\rangle - |\nu_{xj}\rangle\|^2 = \||\mu_{xj}\rangle\|^2 + \||\nu_{xj}\rangle\|^2$. Therefore, optimal solutions to $A_s(F)$ and $B_s(F)$ can be related, in both directions, using $|u_{xj}\rangle = |\mu_{xj}\rangle + |\nu_{xj}\rangle$, $|v_{xj}\rangle = |\mu_{xj}\rangle - |\nu_{xj}\rangle$. ☐

The values of the witness size and general adversary SDPs differ by at most a factor of two, and are equal in the case the function has boolean output:

Theorem 4.2.3. *For finite sets D and E, and $\mathcal{D} \subseteq D^n$, let $f : \mathcal{D} \to E$. The general adversary bound and witness size SDPs are related by*

$$\mathrm{Adv}^{\pm}_s(f) \leq \mathrm{wsize}_s(f) \leq 2\Big(1 - \frac{1}{|E|}\Big)\,\mathrm{Adv}^{\pm}_s(f) \tag{4.2.9}$$

and in particular agree when $|E| = 2$.

Proof. It is clear that $\mathrm{Adv}^{\pm}_s(f) \leq \mathrm{wsize}_s(f)$ as any adversary matrix which is a feasible solution for the general adversary primal SDP will also be feasible for the witness size primal SDP with the same objective value.

For the other direction, use

$$\begin{aligned} \mathrm{wsize}_s(f) &= \max\Big\{\|\Gamma \circ F\| : \forall j \in [n], \|\Gamma \circ \Delta_j\| \leq s_j\Big\} \\ &\leq \max\Big\{\|\Gamma \circ F\| : \forall j \in [n], \|\Gamma \circ \Delta_j \circ F\| \leq \gamma_2(F)\, s_j\Big\} \\ &= \gamma_2(F)\,\mathrm{Adv}^{\pm}_s(f) \ . \end{aligned} \tag{4.2.10}$$

From Eq. (4.2.1), the matrix F satisfies the assumptions of Lemma 4.1.1 with S_b the preimage of $b \in E$. Therefore, $\gamma_2(F) = 2\big(1 - \frac{1}{|E|}\big)$, giving our claim.

It is instructive to give a separate proof based on the dual formulations, as a similar construction will be used both for the algorithm and the composition result. Let $|u_{xj}\rangle, |v_{xj}\rangle$ be a solution to the program $B_s(F)$ *without* the constraints on pairs x, y where $f(x) = f(y)$, and that achieves objective value $\mathrm{Adv}^{\pm}_s(f)$. Let $|\alpha_b\rangle, |\beta_b\rangle$ for $b \in E$ be vectors witnessing that $\gamma_2(G) \leq 2(1 - \frac{1}{|E|})$, where $\langle a|G|b\rangle = 1 - \delta_{a,b}$. In other words, they are the vectors from Eq. (4.1.5) suitably normalized. We claim that $|u_{xj}\rangle \otimes |\alpha_{f(x)}\rangle, |v_{xj}\rangle \otimes |\beta_{f(x)}\rangle$ form a feasible solution to the witness size program $B_s(F)$ given in Eq. (4.2.7), with objective value $\gamma_2(F)\mathrm{Adv}^{\pm}_s(f)$. Indeed, note that $(\langle u_{xj}| \otimes \langle \alpha_{f(x)}|)(|v_{yj}\rangle \otimes |\beta_{f(y)}\rangle) = \langle u_{xj}|v_{yj}\rangle$ if $f(x) \neq f(y)$ and zero otherwise. Furthermore, $\||u_{xj}\rangle \otimes |\alpha_{f(x)}\rangle\| = \sqrt{\gamma_2(F)}\||u_{xj}\rangle\|$ and $\||v_{xj}\rangle \otimes |\beta_{f(x)}\rangle\| = \sqrt{\gamma_2(F)}\||v_{xj}\rangle\|$. □

4.3 Equivalent formulations for witness size

We give few alternate formulations of witness size sdp. They are shown equivalent using simple properties of a semidefinite program. They are useful in motivating the algorithm and deriving composition properties. The witness size program can be written as follows

$$\min_{u_{xi}, v_{xi}} \max_{x} \sum_i \|u_{xi}\|^2 + \|v_{xi}\|^2 \tag{4.3.1}$$
$$\sum_{i: x_i \neq y_i} \langle u_{xi}|u_{yi}\rangle - \langle v_{xi}|v_{yi}\rangle = F[x, y].$$

We propose three different alternative equivalent formulations:

- This is the first alternate SDP for witness size.

$$\min_{\mu_{xi}, \nu_{xi}} \max_{x} \sum_i \|\mu_{xi}\|^2 \tag{4.3.2}$$
$$\sum_{i: x_i \neq y_i} \langle \mu_{xi}|\nu_{yi}\rangle = F[x, y]$$
$$\||\mu_{xi}\rangle\| = \||\nu_{xi}\rangle\|.$$

Proof. Indeed, notice that in the first formulation we may assume without loss of generality that in an optimal solution the $|u_{xi}\rangle$ and $|v_{xi}\rangle$ are orthogonal. Then

let $|\mu_{xi}\rangle = |u_{xi}\rangle + |v_{xi}\rangle$ and $|\nu_{xi}\rangle = |u_{xi}\rangle - |v_{xi}\rangle$ to obtain a solution to the second SDP with the same objective value, since $\langle\mu_{xi}|\nu_{yi}\rangle = \langle u_{xi}|u_{yi}\rangle - \langle v_{xi}|v_{yi}\rangle$ and $\||\mu_{xi}\rangle\|^2 = \||\nu_{xi}\rangle\|^2 = \||u_{xi}\rangle\|^2 + \||v_{xi}\rangle\|^2$. Conversely, beginning with an optimal solution $|\mu_{xi}\rangle, |\nu_{xi}\rangle$ to the second program, let $|u_{xi}\rangle = \frac{1}{2}(|\mu_{xi}\rangle + |\nu_{xi}\rangle)$, $|v_{xi}\rangle = \frac{1}{2}(|\mu_{xi}\rangle - |\nu_{xi}\rangle)$. Then $\sum_i \langle u_{xi}|u_{yi}\rangle - \langle v_{xi}|v_{yi}\rangle = \sum_i(\langle\mu_{xi}|\nu_{yi}\rangle + \langle\mu_{yi}|\nu_{xi}\rangle)/2 = (F[x,y] + F[y,x])/2 = F[x,y]$ and $\||u_{xi}\rangle\|^2 + \||v_{xi}\rangle\|^2 = \||\mu_{xi}\rangle\|^2$. □

- This is the second alternate formulation.

$$A(F) = \min_{\mu_{xi},\nu_{xi}} \max_x \frac{1}{2}\sum_i \|\mu_{xi}\|^2 + \|\nu_{xi}\|^2 \qquad (4.3.3)$$
$$\sum_{i:x_i\neq y_i} \langle\mu_{xi}|\nu_{yi}\rangle = F[x,y].$$

Proof. Let us first see that $A(F)$ 4.3.3 is smaller than the original Formulation 4.3.1. Notice that in the original formulation we may assume without loss of generality that in an optimal solution the u_{xi} and v_{yi} are orthogonal. Let $\{u_{xi}\}, \{v_{yj}\}$ be such a solution. Then let $\mu_{xi} = u_{xi} + v_{xi}$ and $\nu_{xi} = u_{xi} - v_{xi}$. We see that

$$\langle\mu_{xi}|\nu_{yi}\rangle = \langle u_{xi}|u_{yi}\rangle - \langle v_{xi}|v_{yi}\rangle$$

and $\|\mu_{xi}\|^2 = \|\nu_{xi}\|^2 = \|u_{xi}\|^2 + \|v_{xi}\|^2$.

For the other direction, let μ_{xi}, ν_{yi} be an optimal solution to $A(F)$ 4.3.3. Then let $u_{xi} = \frac{1}{2}(\mu_{xi} + \nu_{xi})$ and let $v_{xi} = \frac{1}{2}(\mu_{xi} - \nu_{xi})$. We have

$$\langle\mu_{xi}|\nu_{yi}\rangle = \langle u_{xi}|u_{yi}\rangle - \langle v_{xi}|v_{yi}\rangle$$

and $\|u_{xi}\|^2 + \|v_{xi}\|^2 = \frac{1}{2}(\|\mu_{xi}\|^2 + \|\nu_{xi}\|^2)$. □

- We can now write down third alternative formulation where the objective function is a sum of products.

$$B(F) = \min_{\mu_{xi},\nu_{xi}} \max_x \sum_i \|\mu_{xi}\|\|\nu_{xi}\| \qquad (4.3.4)$$
$$\sum_{i:x_i\neq y_i} \langle\mu_{xi}|\nu_{yi}\rangle = F[x,y].$$

Claim 4.3.1. *The quantities $A(F)$ and $B(F)$ defined above are equal.*

$$A(F) = B(F)$$

Proof. By the arithmetic-geometric mean inequality it is clear that $B(F) \leq A(F)$. To complete the proof, we will show that $B(F)$ is at least as large as $A(F)$ by showing that it is at least as large as the *primal* program of $A(F)$. As we already know that $A(F)$ is equal to witness size, the primal program is

$$\max \left\{ \|\Gamma \circ F\| : \forall j \in [n], \|\Gamma \circ \Delta_j\| \leq 1 \right\}$$

The condition $\|\Gamma \circ \Delta_j\| \leq 1$ implies that

$$\Delta_i = \begin{bmatrix} I & \Gamma \circ D_j \\ \Gamma \circ D_j & I \end{bmatrix} \succeq 0$$

for each j.

Suppose that we have a decomposition of F as $F = \sum_j X_j Y_j^* \circ D_j$. Form the positive semidefinite matrix

$$P_i = \begin{bmatrix} X_i X_i^* & X_i Y_i^* \\ Y_i X_i^* & Y_i Y_i^* \end{bmatrix}.$$

As the Hadamard product of two psd matrices is psd, the matrix $\sum_i \Delta_i \circ P_i$ is also psd. Now we use the following lemma (see [Bha07], Positive Definite Matrices, Prop. 1.3.2).

Lemma 4.3.2. *If*

$$\begin{bmatrix} A & X \\ X^* & B \end{bmatrix} \succeq 0$$

then $\|X\| \leq \|A\|^{1/2} \|B\|^{1/2}$.

Applying this to the matrix $\sum_i \Delta_i \circ P_i$ we find that

$$\|\sum_i \Gamma \circ D_i \circ X_i Y_i^*\| = \|\Gamma \circ F\| \leq \|\sum_i X_i X_i^* \circ I\|^{1/2} \|\sum_i Y_i Y_i^* \circ I\|^{1/2}.$$

This gives the claim. ∎

4.4 The general adversary bound is tight

In this section, we will prove Theorem 4.0.1 by exhibiting an algorithm that evaluates f. As sketched in Figure 4.1, the idea is to run a quantum walk on a certain graph constructed using the witness size SDP. The first step is to turn a solution to the dual SDP formulation given in Eq. (4.2.7) into a more natural geometric object. If the input alphabet is $D = [k]$, we do this by combining a witness size solution for f with a γ_2 solution for the matrix G given by $\langle a|G|b\rangle = 1 - \delta_{a,b}$ for $a, b \in [k]$.

Indeed let $|\mu_i\rangle, |\nu_i\rangle$ be the vectors given in Eq. (4.1.5), i.e., an optimal γ_2 factorization of G up to normalization. Notice that we can rewrite the sum

$$\sum_{j\in[n]:\, x_j\neq y_j} \langle u_{xj}|v_{yj}\rangle = \frac{2(k-1)}{k} \sum_{j\in[n]} \langle u_{xj}|v_{xj}\rangle\langle \mu_{x_j}|\nu_{y_j}\rangle \ . \tag{4.4.1}$$

The point of this transformation is that now the inner product between the vectors $\sum_j |j\rangle|u_{xj}\rangle|\mu_{x_j}\rangle$ and $\frac{2(k-1)}{k}\sum_j |j\rangle|v_{yj}\rangle|\nu_{y_j}\rangle$ will equal $\langle x|F|y\rangle$. The former vectors will be used to create our weighted graph, and all the vectors will be used to construct eigenvalue-zero eigenvectors of the graph. The constraint $\sum_{j:\, x_j\neq y_j} \langle u_{xj}|v_{xj}\rangle = \langle x|F|y\rangle$ will be required for all input pairs $x, y \in \mathcal{D}$.

4.4.1 The algorithm

Let $f : \mathcal{D} \to E$, with $\mathcal{D} \subseteq [k]^n$. Consider the dual formulation of witness size given by Eq. (4.2.7) with uniform costs $s = \vec{1}$. Based on a feasible solution to this SDP with objective value W, we will give an algorithm for evaluating f with query complexity $O(W)$. Note that $W \geq 1$ necessarily.

We first construct a weighted bipartite graph G using the vectors $|u_{xj}\rangle, |v_{xj}\rangle \in \mathbf{C}^m$ from an optimal solution to $B_{\vec{1}}(F)$, and the vectors $|\mu_i\rangle, |\nu_i\rangle \in \mathbf{C}^k$ defined in Lemma 4.1.1, Eq. (4.1.5). The graph G is between $1 + |\mathcal{D}|$ vertices, which we label by the elements of $\{c\} \cup \mathcal{D}$, and $1 + |E| + knm$ vertices, which we label by $\{\emptyset\} \cup E \cup I$, where $I = [n] \times [k] \times [m]$.

Let $A \in \mathcal{L}(\mathbf{C}^I, \mathbf{C}^{\{c\}\cup\mathcal{D}})$ be given by

$$A = \sum_{x\in\mathcal{D},\, j\in[n]} |x\rangle\langle j| \otimes \langle \mu_{x_j}| \otimes \langle u_{xj}| \ . \tag{4.4.2}$$

By extending it with zeros, A can also be seen to act on $\mathbf{C}^{\{\emptyset\}\cup E\cup I} \supset \mathbf{C}^I$. Let $\eta = 1/100$, and define vectors $|c_b\rangle \in \mathbf{C}^{\{c\}\cup\mathcal{D}}$, for $b \in E$, by

$$|c_b\rangle = -|c\rangle - \frac{\eta}{\sqrt{W}} \sum_{x\in\mathcal{D}: f(x)\neq b} |x\rangle \ . \tag{4.4.3}$$

Let $V = (\{c\} \cup \mathcal{D}) \cup (\{\emptyset\} \cup E \cup I)$, and let G be the weighted bipartite graph with vertex set V and biadjacency matrix $B_G \in \mathcal{L}(\mathbf{C}^{\{\emptyset\}\cup E\cup I}, \mathbf{C}^{\{c\}\cup\mathcal{D}})$:

$$B_G = |c\rangle\langle\emptyset| + \sum_{b\in E} |c_b\rangle\langle b| + A \ . \tag{4.4.4}$$

An example is shown in Figure 4.1, although for ease of illustration the example uses the first set of vectors $|\mu_i\rangle, |\nu_i\rangle$ from the proof of Lemma 4.1.1, and not those from Eq. (4.1.5).

Let $\Delta \in \mathcal{L}(\mathbf{C}^V)$ be the orthogonal projection onto the span of all eigenvalue-zero eigenvectors of the weighted adjacency matrix A_G. For an input $x \in \mathcal{D}$, let $\Pi_x \in \mathcal{L}(\mathbf{C}^V)$ be the projection

$$\Pi_x = \mathbf{1} - \sum_{j\in[n]} |j\rangle\langle j| \otimes |\mu_{x_j}\rangle\langle\mu_{x_j}| \otimes \mathbf{1}_{\mathbf{C}^m} \ . \tag{4.4.5}$$

Finally, let

$$U_x = (2\Pi_x - \mathbf{1})(2\Delta - \mathbf{1}) \ . \tag{4.4.6}$$

U_x consists of the two reflections $2\Delta - \mathbf{1}$ and $2\Pi_x - \mathbf{1}$. The first reflection does not depend on the input x. The second reflection can be implemented using a call to the input oracle O_x and its inverse: compute x_j, reflect $|\mu_{x_j}\rangle$, then uncompute x_j.

Our algorithm is as follows:

Algorithm:

1. Let $\tau = \lceil 10^9 W \rceil$. Prepare the initial state $\frac{1}{\sqrt{\tau}} \sum_{t\in[\tau]} |t\rangle \otimes |\emptyset\rangle \in \mathbf{C}^{[\tau]} \otimes \mathbf{C}^V$.
2. Apply the controlled unitary $\sum_{t\in[\tau]} |t\rangle\langle t| \otimes U_x^t$.
3. By a measurement, project onto the span of the states $\frac{1}{\sqrt{\tau}} \sum_{t\in[\tau]} |t\rangle \otimes |b\rangle$, for $b \in E$. On success, output the second register. Otherwise, fail.

4.4.2 Analysis of the algorithm

Proposition 4.4.1. *The algorithm outputs $f(x)$ with probability at least 21%. For any $b \in E \setminus \{f(x)\}$, the algorithm outputs b with probability at most 0.2%.*

The algorithm makes $\tau = O(W)$ queries to O_x. Theorem 4.0.1 therefore follows from Proposition 4.4.1, by using repetition to improve the success rate. To prove Proposition 4.4.1, we shall study the spectrum of the unitary U_x. The structure of the argument is similar to the analysis in[Rei10c].

Let

$$|\phi_\pm\rangle = \frac{1}{\sqrt{2}}\big(|\emptyset\rangle \pm |f(x)\rangle\big) \ . \tag{4.4.7}$$

For the analysis, it will be useful to introduce two new graphs. Let $\overline{\Pi}(x) = \sum_{j\in[n]} |j\rangle\langle j| \otimes |\mu_{x_j}\rangle\langle\mu_{x_j}| \otimes \mathbf{1}_{\mathbf{C}^m}$, considered as an element of $\mathcal{L}(\mathbf{C}^{\{\emptyset\}\cup E\cup I}, \mathbf{C}^I)$, and let $G(x)$ and $G'(x)$ be the weighted bipartite graphs with biadjacency matrices

$$B_{G(x)} = \begin{pmatrix} B_G \\ \overline{\Pi}(x) \end{pmatrix} \qquad \text{and} \qquad B_{G'(x)} = B_{G(x)}(\mathbf{1} - |\phi_-\rangle\langle\phi_-|) \ . \tag{4.4.8}$$

The extra $\overline{\Pi}(x)$ block in the $B_{G(x)}$ matrix has the effect of attaching disjoint bipartite graphs to G's column vertices, with the graphs on vertices labeled j each having biadjacency matrix $|\mu_{x_j}\rangle\langle\mu_{x_j}|$. For example, if the input alphabet is binary, $k = 2$, then observe from Eq. (4.1.5) that $|\mu_{x_j}\rangle\langle\mu_{x_j}| = |\bar{x}_j\rangle\langle\bar{x}_j|$, so $G(x)$ differs from G by the attachment of dangling weight-one edges to all vertices $|j, \bar{x}_j, i\rangle$ for $j \in [n], i \in [m]$, i.e., all vertices with a label that disagrees with the input string x.

Based on the constraints of the SDP in Eq. (4.2.7), we can construct eigenvalue-zero eigenvectors for $G(x)$ and $G'(x)$:

Lemma 4.4.2. *The vector*

$$|\psi\rangle = |\emptyset\rangle + |f(x)\rangle + \frac{\eta}{\sqrt{W}} \frac{2(k-1)}{k} \sum_{j\in[n]} |j\rangle \otimes |\nu_{x_j}\rangle \otimes |v_{xj}\rangle \tag{4.4.9}$$

satisfies $B_{G(x)}|\psi\rangle = 0$ *and* $|\langle\phi_+|\psi\rangle|^2/\||\psi\rangle\|^2 > 1 - 2\eta^2 = 1 - 2\cdot 10^{-4}$.

The vector

$$|\psi'\rangle = -\frac{\eta}{\sqrt{W}}|c\rangle + |x\rangle - \sum_{j\in[n]} |j\rangle \otimes |\mu_{x_j}\rangle \otimes |u_{xj}\rangle \tag{4.4.10}$$

satisfies $B^\dagger_{G'(x)}|\psi'\rangle = 0$ *and* $|\langle\phi_-|B^\dagger_{G(x)}|\psi'\rangle|^2/\||\psi'\rangle\|^2 > 9/(10^5 W^2)$.

Proof. By definition of $B_{G(x)}$, we have $B_{G(x)}|\psi\rangle = (B_G|\psi\rangle, \overline{\Pi}(x)|\psi\rangle)$. The first term is

$$\begin{aligned} B_G|\psi\rangle &= \Big(|c\rangle\langle\emptyset| + \sum_{b\in E}|c_b\rangle\langle b| + \sum_{y\in\mathcal{D},j\in[n]} |y\rangle\langle j| \otimes \langle\mu_{y_j}| \otimes \langle u_{yj}|\Big)|\psi\rangle \\ &= |c\rangle + |c_{f(x)}\rangle + \frac{\eta}{\sqrt{W}}\frac{2(k-1)}{k}\sum_{y\in\mathcal{D},j\in[n]} |y\rangle\langle\mu_{y_j}|\nu_{x_j}\rangle\langle u_{yj}|v_{xj}\rangle \\ &= |c\rangle + |c_{f(x)}\rangle + \frac{\eta}{\sqrt{W}}\sum_{y\in\mathcal{D}}\sum_{j\in[n]:y_j\neq x_j} |y\rangle\langle u_{yj}|v_{xj}\rangle \\ &= 0 \ , \end{aligned} \tag{4.4.11}$$

where in the third equation we used $\langle\mu_{y_j}|\nu_{x_j}\rangle$ is 0 if $x_j = y_j$ and is otherwise $k/(2(k-1))$, and in the last step we used the SDP constraint $\sum_{j:x_j\neq y_j}\langle u_{yj}|v_{xj}\rangle = 1 - \delta_{f(x),f(y)}$ and the definition of $|c_{f(x)}\rangle$. The second term, $\overline{\Pi}(x)|\psi\rangle$, evaluates to zero since $\langle\mu_{x_j}|\nu_{x_j}\rangle = 0$:

$$\begin{aligned} \overline{\Pi}(x)|\psi\rangle &= \frac{\eta}{\sqrt{W}}\frac{2(k-1)}{k}\overline{\Pi}(x)\sum_{j\in[n]} |j\rangle \otimes |\nu_{x_j}\rangle \otimes |v_{xj}\rangle \\ &= \frac{\eta}{\sqrt{W}}\frac{2(k-1)}{k}\sum_{j\in[n]}\langle\mu_{x_j}|\nu_{x_j}\rangle|j\rangle \otimes |\mu_{x_j}\rangle \otimes |v_{xj}\rangle = 0 \ . \end{aligned} \tag{4.4.12}$$

Thus indeed $B_{G(x)}|\psi\rangle = 0$. The claim $|\langle\phi_+|\psi\rangle|^2/\||\psi\rangle\|^2 \geq \frac{2}{2+4\eta^2} > 1 - 2\eta^2$ is a calculation, using $\||\nu_{x_j}\rangle\| = 1$ and $\sum_j \||v_{xj}\rangle\|^2 \leq W$.

For the second part of the lemma, we claim that $B^\dagger_{G'(x)}|\psi'\rangle = 0$. Indeed,

$$B^\dagger_{G'(x)}|\psi'\rangle = (\mathbf{1} - |\phi_-\rangle\langle\phi_-|)\begin{bmatrix} B_G^\dagger & \overline{\Pi}(x)\end{bmatrix}\begin{bmatrix} -\frac{\eta}{\sqrt{W}}|c\rangle + |x\rangle \\ -\sum_j |j\rangle \otimes |\mu_{x_j}\rangle \otimes |u_{xj}\rangle\end{bmatrix} \tag{4.4.13}$$

$$\begin{aligned} &= (\mathbf{1} - |\phi_-\rangle\langle\phi_-|)\Big\{\Big(|\emptyset\rangle\langle c| + \sum_{b\in E}|b\rangle\langle c_b|\Big)\Big(\frac{-\eta}{\sqrt{W}}|c\rangle + |x\rangle\Big) \\ &\quad + A^\dagger|x\rangle - \overline{\Pi}(x)\sum_{j\in[n]}|j\rangle|\mu_{x_j}\rangle|u_{xj}\rangle\Big\} \end{aligned} \tag{4.4.14}$$

The term $A^\dagger|x\rangle$ cancels the $\overline{\Pi}(x)$ term, while the other terms are proportional to $|\phi_-\rangle$:

$$\begin{aligned}\big(|\emptyset\rangle\langle c| + \sum_{b\in E}|b\rangle\langle c_b|\big)\big(\frac{-\eta}{\sqrt{W}}|c\rangle + |x\rangle\big) &= \Big(\frac{-\eta}{\sqrt{W}}\big(|\emptyset\rangle - \sum_{b\in E}|b\rangle\big) - \frac{\eta}{\sqrt{W}}\sum_{b:f(x)\neq b}|b\rangle\Big) \\ &= \frac{-\eta}{\sqrt{W}}\big(|\emptyset\rangle - |f(x)\rangle\big) = \frac{-\eta}{\sqrt{W}}\sqrt{2}|\phi_-\rangle \ .\end{aligned} \tag{4.4.15}$$

Finally, use $B_{G(x)}|\phi_-\rangle = \frac{1}{\sqrt{2}}\big(2|c\rangle + \frac{\eta}{\sqrt{W}}\sum_{y\in\mathcal{D}:f(y)\neq f(x)}|y\rangle\big)$ to calculate $|\langle\phi_-|B^\dagger_{G(x)}|\psi'\rangle|^2/\||\psi'\rangle\|^2 \geq \frac{2\eta^2}{W}/(\frac{\eta^2}{W}+1+W) > 9/(10^5W^2)$. □

Our main technical tool is the following theorem, abstracted from the analysis in[Rei10c]:

Theorem 4.4.3 ([Rei10c])**.** *Let G be a weighted bipartite graph with biadjacency matrix $B_G \in \mathcal{L}(\mathbf{C}^S, \mathbf{C}^T)$. Let $\Delta \in \mathcal{L}(\mathbf{C}^S \oplus \mathbf{C}^T)$ be the orthogonal projection onto the span of all eigenvalue-zero eigenvectors of the weighted adjacency matrix A_G. Let $S_0 \subset S$, and define G_0 from G by adding a dangling, weight-one edge to each vertex in S_0. Let Π_0 be a diagonal matrix that projects onto all vertices except those in S_0. Let $U_0 = (2\Pi_0 - \mathbf{1})(2\Delta - \mathbf{1})$, and let $\{|\beta\rangle\}$ be a complete set of orthonormal eigenvectors of U_0 with corresponding eigenvalues $e^{i\theta(\beta)}$, $\theta(\beta) \in (-\pi, \pi]$. Then,*

- *If there are unit-length vectors $|\psi\rangle, |\phi\rangle \in \mathbf{C}^S$ such that $|\phi\rangle$ is not supported on S_0, $B_{G_0}|\psi\rangle = 0$ and $|\langle\phi|\psi\rangle|^2 \geq \delta$, then U_0 has an unit-length, eigenvalue-one eigenvector $|\beta_0\rangle$ with $|\langle\beta_0|\phi\rangle|^2 \geq \delta$.*

- *If there are unit-length vectors $|\psi'\rangle \in \mathbf{C}^T \oplus \mathbf{C}^{S_0}, |\phi'\rangle \in \mathbf{C}^S$ such that $|\phi'\rangle$ is not supported on S_0, $(\mathbf{1} - |\phi'\rangle\langle\phi'|)B^\dagger_{G_0}|\psi'\rangle = 0$ and $|\langle\phi'|B^\dagger_{G_0}|\psi'\rangle|^2 \geq \delta$, then for any $\Theta \geq 0$,*

$$\sum_{\beta:|\theta(\beta)|\leq\Theta}|\langle\beta|\phi'\rangle|^2 \leq \Big(2\sqrt{\delta\Theta} + \frac{\Theta}{2}\Big)^2 \ . \tag{4.4.16}$$

Our graphs do not quite satisfy the hypotheses of the theorem, since going from G to $G(x)$ involves adding edges weighted by the entries of $|\mu_{x_j}\rangle\langle\mu_{x_j}|$ for $j \in [n]$, and not weight-one edges. However, since $\||\mu_i\rangle\| = 1$, we can apply a unitary change of basis so that our graphs and the operator U_x are in accord with the theorem. Combining Lemma 4.4.2 and Theorem 4.4.3, we obtain:

Lemma 4.4.4. *U_x has a unit-length, eigenvalue-one eigenvector $|\beta_0\rangle$ with*

$$|\langle\beta_0|\phi_+\rangle|^2 > 1 - 2\cdot 10^{-4} \ . \tag{4.4.17}$$

This eigenvector is simply the restriction of $|\psi\rangle$ from Eq. (4.4.9) to the vertices of G.

Furthermore, let $\{|\beta\rangle\}$ be a complete set of orthonormal eigenvectors of U_x with corresponding eigenvalues $e^{i\theta(\beta)}$, $\theta(\beta) \in (-\pi, \pi]$. Then for any $\Theta \geq 0$,

$$\sum_{\beta:|\theta(\beta)|\leq\Theta} |\langle\beta|\phi_-\rangle|^2 \leq \left(10\sqrt{6\Theta W} + \frac{\Theta}{2}\right)^2 \ . \tag{4.4.18}$$

There is now clear intuition for Proposition 4.4.1. Note that $|\emptyset\rangle = \frac{1}{\sqrt{2}}\big(|\phi_+\rangle + |\phi_-\rangle\big)$. The first term, $|\phi_+\rangle$, is very close to an eigenvalue-zero eigenvector of $A_{G(x)}$, while the second term, $|\phi_-\rangle$, is very far from the small-eigenvalue subspace of $A_{G(x)}$. Intuitively, this allows us to average out the $|\phi_-\rangle$ term. The surviving first term, then, has constant overlap on the desired vertex, $|f(x)\rangle$. A full proof is given in Appendix B.1.

4.5 Composition of the general adversary and witness size SDPs

We now study the behavior of the two SDPs under function composition.

Let us begin by introducing some notation. For functions $g_i : \mathcal{C}_i \to D$, with $\mathcal{C}_i \subseteq C^{m_i}$, let $M = \sum_{i\in[n]} m_i$ and define a function $\vec{g} = (g_1, \ldots, g_n)$ mapping $\mathcal{C}_1 \times \cdots \times \mathcal{C}_n \subseteq C^M$ to D^n by

$$\vec{g}(x) = \big(g_1(x_1, \ldots, x_{m_1}), \ldots, g_n(x_{M-m_n+1}, \ldots, x_M)\big) \ . \tag{4.5.1}$$

For the purposes of query complexity and the general adversary bound and witness size SDPs, the decomposition of the domain is into copies of C. In the special case that the functions g_i are the same g, we write $g^n = \vec{g}$.

Lemma 4.5.1. *Let $f : D^n \to E$, and for $i \in [n]$ let $g_i : \mathcal{C}_i \to D$, $\mathcal{C}_i \subseteq C^{m_i}$. Let $s_i = \mathrm{wsize}(g_i)$. Then*

$$\mathrm{Adv}^{\pm}(f \circ \vec{g}) \leq \mathrm{Adv}^{\pm}_{\vec{s}}(f) \tag{4.5.2}$$

$$\mathrm{wsize}(f \circ \vec{g}) \leq \mathrm{wsize}_{\vec{s}}(f) \ . \tag{4.5.3}$$

The proof of this lemma follows in the natural way. We take optimal dual solutions to the witness size programs for f and the g_i, using the formulation of Eq. (4.2.7), and form their tensor product to construct a solution to the composed program. This proof strategy does not directly work for the general adversary bound—we crucially use the extra constraints present in the witness size programs for the g_i.

Proof of Lemma 4.5.1. An input x to $f \circ \vec{g}$ can be written as $x = (x^1, \ldots, x^n)$, with $x^i \in \mathcal{C}_i$. Let $\tilde{x} = \vec{g}(x) \in D^n$, i.e., $\tilde{x}_i = g_i(x^i)$ for $i \in [n]$.

Let us begin with showing Eq. (4.5.3). For $i \in [n]$, let $\{|\mu_{x^i,j}\rangle, |\nu_{x^i,j}\rangle\}$ be optimal vectors for g_i in Eq. (4.2.7) with uniform costs, and let $\{|u_{\tilde{x},i}\rangle, |v_{\tilde{x},i}\rangle\}$ be optimal vectors for f with costs s.

We design a vector solution to Eq. (4.2.7) for the composed function $f \circ \vec{g}$ as

$$|\alpha_{x,(i,j)}\rangle = |u_{\tilde{x},i}\rangle \otimes |\mu_{x^i,j}\rangle \qquad |\beta_{x,(i,j)}\rangle = |v_{\tilde{x},i}\rangle \otimes |\nu_{x^i,j}\rangle \ . \tag{4.5.4}$$

Then $\||\alpha_{x,(i,j)}\rangle\| = \||u_{\tilde{x},i}\rangle\|\||\mu_{x^i,j}\rangle\|$ and $\||\beta_{x,(i,j)}\rangle\| = \||v_{\tilde{x},i}\rangle\|\||\nu_{x^i,j}\rangle\|$, and for all inputs x, y,

$$\begin{aligned}
\sum_{(i,j):x^i_j \neq y^i_j} \langle \alpha_{x,(i,j)} | \beta_{y,(i,j)} \rangle &= \sum_{(i,j):x^i_j \neq y^i_j} \langle u_{\tilde{x},i} | v_{\tilde{x},i} \rangle \langle \mu_{x^i,j} | \nu_{x^i,j} \rangle \\
&= \sum_i \langle u_{\tilde{x},i} | v_{\tilde{y},i} \rangle \sum_{j:x^i_j \neq y^i_j} \langle \mu_{x^i,j} | \nu_{y^i,j} \rangle \\
&= \sum_{i:\tilde{x}_i \neq \tilde{y}_i} \langle u_{\tilde{x},i} | v_{\tilde{y},i} \rangle \\
&= 1 - \delta_{f(\tilde{x}),f(\tilde{y})} \ .
\end{aligned} \tag{4.5.5}$$

The key point here is that the witness size (or adversary) SDP for $f \circ \vec{g}$ imposes a constraint on the summation over all i, j with $x^i_j \neq y^i_j$. This includes those i, j such that $x^i_j \neq y^i_j$ yet $\tilde{x}_i = g_i(x^i) = \tilde{y}_i = g_i(y^i)$. With the adversary bound we have no control over the inner products in this case. The witness size SDP for g_i, on the other hand, gives that the sums are zero when $\tilde{x}_i = \tilde{y}_i$.

Thus we upper bound the witness size for the composed function $f \circ \vec{g}$ by the objective value, the larger of $\max_x \sum_{i,j} \||\alpha_{x,(i,j)}\rangle\|^2$ and $\max_x \sum_{i,j} \||\beta_{x,(i,j)}\rangle\|^2$. Both

expressions can be bounded in the same way, e.g.,

$$\sum_{i\in[n],j\in[m_i]} \||\alpha_{x,(i,j)}\rangle\|^2 \leq \sum_{i\in[n]} \|u_{\tilde{x},i}\|^2 \operatorname{wsize}(g_i) \leq \operatorname{wsize}_s(f) \ . \tag{4.5.6}$$

By Eq. (4.2.9), this also implies Eq. (4.5.2), except with a lost factor of two on the right-hand side. That this factor of two is unnecessary can be seen easily by repeating the above arguments, except beginning with an optimal vector solution to the general adversary bound dual SDP for f from Lemma 4.2.2, and then considering only inputs x, y with $(f \circ \vec{g})(x) \neq (f \circ \vec{g})(y)$. □

Corollary 4.5.2. *Let $g : \mathcal{C} \to D$, with $\mathcal{C} \subseteq C^m$, and let $f : D^n \to E$. Then*

$$\operatorname{Adv}^{\pm}(f \circ g^n) \leq \operatorname{Adv}^{\pm}(f) \operatorname{wsize}(g) \tag{4.5.7}$$

$$\operatorname{wsize}(f \circ g^n) \leq \operatorname{wsize}(f) \operatorname{wsize}(g) \ . \tag{4.5.8}$$

In the case where all the functions f and g_i are boolean, a matching lower bound to Lemma 4.5.1 has been shown by Høyer et al. [HLŠ07]. Such a composition lower bound cannot hold in general for non-boolean functions. For example, let $f : [k]^n \to \mathbf{B}$ output the sum of its inputs modulo two. Its quantum query complexity $Q(f)$ is $\Theta(n)$. Let $g : \mathbf{B}^m \to [k]$ be a function with $Q(g) = \Theta(m)$ but that only outputs even numbers. Then the composition $f \circ g^n$ is the constant zero function.

A matching composition lower bound does hold in the case that the range of g is boolean:

Lemma 4.5.3. *Let $f : \{0,1\}^n \to E$, and, for $i \in [n]$, $g_i : \mathcal{C}_i \to \{0,1\}$. Let $s_i = \operatorname{wsize}(g_i)$. Then*

$$\operatorname{wsize}(f \circ \vec{g}) \geq \operatorname{wsize}_s(f) \ . \tag{4.5.9}$$

Proof. The proof follows the same lines as the proof for the boolean case given in [HLŠ07].

Our basic task is, given optimal matrices Γ_f, Γ_{g_i} from the witness size primal SDPs, construct a matrix Γ to show that the witness size of $f \circ \vec{g}$ is large. Note that as each g_i has boolean output, we may assume by Theorem 4.2.3 that the Γ_{g_i} are adversary matrices.

A key part of this construction is a *block-wise Hadamard product*. Let $B_1, \ldots, B_n$ be n matrices, with each B_i structured by arbitrary divisions of the rows and of the columns as

$$B_i = \begin{bmatrix} B_i^{0,0} & B_i^{0,1} \\ B_i^{1,0} & B_i^{1,1} \end{bmatrix} \quad . \tag{4.5.10}$$

Now the tensor product $\otimes B_i$ will have a natural block structure where $(x, y) \in \{0,1\}^n \times \{0,1\}^n$ labels the block $\otimes B_i^{x_i, y_i}$. For a 2^n-by-2^n matrix A, define the block-wise Hadamard product $A \circ_b (\bigotimes B_i)$ as the matrix $\otimes B_i$ with block x, y multiplied by $A[x, y]$.

Claim 4.5.4. *Let A be symmetric and B_i be as above with $B_i^{1,0} = (B_i^{0,1})^\dagger$ and $B_i^{0,0} = \|B_i^{0,1}\|\mathbf{1}$, $B_i^{1,1} = \|B_i^{0,1}\|\mathbf{1}$, where $\mathbf{1}$ in each case is an identity matrix of the appropriate dimension. Then*

$$\left\| A \circ_b \left(\bigotimes B_i \right) \right\| = \|A\| \prod_i \left\| B_i^{0,1} \right\| \quad . \tag{4.5.11}$$

Note that $\|B_i\| = 2\|B_i^{0,1}\|$ so the naive upper bound would be $2^n \|A\| \prod_i \left\|B_i^{0,1}\right\|$. We give the proof of this claim in Appendix B.2.

Now define $\Gamma = \Gamma_f \circ_b \left(\bigotimes(\Gamma_{g_i} + \|\Gamma_{g_i}\|\mathbf{1})\right)$. As the Γ_{g_i} are adversary matrices, the rest of the proof follows as in [HLŠ07, Lemma 14], using Claim 4.5.4. □

Corollary 4.5.5. *Let $g : \mathcal{C} \to \{0,1\}$ and $f : \{0,1\}^n \to E$. Then*

$$\operatorname{wsize}(f \circ g^n) = \operatorname{wsize}(f) \operatorname{wsize}(g) \quad . \tag{4.5.12}$$

By Theorem 4.0.1, we obtain:

Corollary 4.5.6. *Let $g : \mathcal{C} \to D$ and $f : D^n \to E$. Then*

$$Q(f \circ g^n) = O\big(Q(f)\, Q(g)\big) \tag{4.5.13}$$

and $Q(f \circ g^n) = \Omega\big(Q(f)Q(g)\big)$ if $|D| = 2$.

The above composition lemmas also lead to direct-sum results for quantum query complexity. In the special case where f is the identity function $D^n \to D^n$, $f \circ \vec{g} = \vec{g}$, and it can be verified that $\operatorname{wsize}_s(f) \leq \sum_j s_j$. Lemma 4.5.1 shows $\operatorname{wsize}(g^n) \leq n \operatorname{wsize}(g)$ and $\operatorname{Adv}^{\pm}(g^n) \leq 2n \operatorname{Adv}^{\pm}(g)$. Ambainis et al. [ACGT10] have shown the

corresponding lower bound $n \operatorname{Adv}^{\pm}(g) \leq \operatorname{Adv}^{\pm}(g^n)$, and their construction can also be used to obtain $n \operatorname{wsize}(g) \leq \operatorname{wsize}(g^n)$. Putting this together, we have the following immediate corollary:

Corollary 4.5.7. *Let $f : \mathcal{D} \to E$, and let $f^n : \mathcal{D}^n \to E^n$ consist of n independent copies of f, given by $f^n(x^1, \ldots, x^n) = \big(f(x^1), \ldots, f(x^n)\big)$. Then*

$$Q(f^n) = \Theta\big(n\, Q(f)\big) \ . \tag{4.5.14}$$

Let us remark that when $E = \{0, 1\}$, the upper bound $Q(f^n) = O\big(n\, Q(f)\big)$ follows from the robust input recovery quantum algorithm [BNRW05, Theorem 3]. The same algorithm can be generalized to handle larger E.

A tight direct-sum property also holds for certificate complexity (nondeterministic classical query complexity). Jain et al. [JKS10] have recently shown a tight direct-sum result for deterministic decision-tree complexity (classical query complexity), and a potentially slightly loose direct-sum result in the randomized case. Their results hold for relations, and not only for functions.

Chapter 5

Optimization over the copositive cone

In this chapter, we look at the application of another kind of convex programming, called copositive programming. It is very similar to semidefinite programming, the difference being that we optimize over the copositive cone instead of the semidefinite cone. We know that the semidefinite cone is the cone of $n \times n$ matrices M, s.t. $\forall\ v \in \mathbb{R}^n,\ \ v^T M v \geq 0$. A matrix is copositive (in the copositive cone), if $\forall\ v \geq 0 \in \mathbb{R}^n,\ \ v^T M v \geq 0$. So the difference is that the quadratic form should be positive only for entry wise positive vectors.

We know that the dual of the semidefinite cone is the semidefinite cone itself. The dual of the copositive cone, on the other hand, is called the completely positive cone. So copositive programming and completely positive programming are equivalent. A matrix is completely positive if $\exists\ v_1, \cdots, v_k \geq 0 \in \mathbb{R}^n,\ \ M = v_1 v_1^T + \cdots + v_k v_k^T$. If we remove the constraint that v_i's are entry wise positive then we get the semidefinite cone. This means that a solution to copositive programming (completely positive programming) can be interpreted as vectors which are entry wise positive. We will call such vectors "completely positive". This notion is helpful in the rounding procedure discussed in Section 2.4.

In general, copositive (completely positive) programming is NP-complete. For most cases, we convert the SDP solution to integer solution directly. But it seems that for certain Semidefinite Programming relaxations, converting the vector solution to a completely positive solution and then using this solution to generate the integer solution is easier. We take a look at the properties of this completely positive solution and study an example where it is useful.

5.1 Notation

Let us consider an integer program in variables s_i ($i \in [n]$):

$$\begin{aligned} \min \quad & \sum_{i,j\in[n]} J_{i,j}\langle s_i|s_j\rangle && (5.1.1)\\ s.t. \quad & \forall k \quad \sum_{i,j\in[n]} \mathbf{A}_{i,j,k} s_i s_j = 1 \\ & \forall k \quad \sum_{i,j\in[n]} \mathbf{B}_{i,j,k} s_i s_j = 0 \quad \forall i \neq j \\ & \forall i \ s_i \in \{0,1\} && (5.1.2) \end{aligned}$$

Its relaxation, when integers s_i are relaxed to vectors u_i, will look like:

$$\begin{aligned} \min \quad & \sum_{i,j\in[n]} J_{i,j}\langle u_i|u_j\rangle && (5.1.3)\\ s.t. \quad & \forall k \quad \sum_{i,j\in[n]} \mathbf{A}_{i,j,k}\langle u_i|u_j\rangle = 1 \\ & \forall k \quad \sum_{i,j\in[n]} \mathbf{B}_{i,j,k}\langle u_i|u_j\rangle = 0 \quad \forall i \neq j \end{aligned}$$

This will be called a semidefinite instance $\pi = (J, \mathbf{A}, \mathbf{B})$. Here $\mathbf{A}, \mathbf{B}$ are the constraint matrix, and matrix $\mathbf{B}$ can be used to make selected dot products zero. Matrix J is known as the objective matrix. Then a completely positive solution to this program will be a solution of the same SDP except one additional constraint of all vectors being positive.

$$\begin{aligned} \min \quad & \sum_{i,j\in[n]} J_{i,j}\langle v_i|v_j\rangle && (5.1.4)\\ s.t. \quad & \forall k \quad \sum_{i,j\in[n]} \mathbf{A}_{i,j,k}\langle v_i|v_j\rangle = 1 \\ & \forall k \quad \sum_{i,j\in[n]} \mathbf{B}_{i,j,k}\langle v_i|v_j\rangle = 0 \quad \forall i \neq j \\ & \forall \ i \in [n] \quad v_i \geq 0 \quad \text{(entry wise)} \end{aligned}$$

Notice that this optimization problems is not an SDP because of the new constraints. It is now a copositive programming problem. Given an SDP (5.1.3) we will denote its solution by $\{u_i\}$, and its completely positive solution (5.1.4) by $\{v_i\}$. Let's denote the solution of original problem (5.1.1) as $\{s_i\}$.

5.2 Gaps between solutions

In rounding an SDP solution, the focus is to bound the gap between the SDP solution and its rounded integer solution. So we want to give an upper bound on the quantity

$$\frac{\sum_{i,j\in[n]} J_{i,j}\langle u_i|u_j\rangle}{\sum_{i,j\in[n]} J_{i,j}\langle s_i|s_j\rangle}, \tag{5.2.1}$$

i.e., the ratio between the objective value of $\{u_i\}$ solution and $\{s_i\}$ solution. With the intermediate step of solving a copositive programming problem, there are two gaps now. We need to bound the gap between the SDP solution and the completely positive solution, and then also bound the gap between the completely positive solution and the integer solution. The two gaps are

1. $\frac{\sum_{i,j\in[n]} J_{i,j}\langle u_i|u_j\rangle}{\sum_{i,j\in[n]} J_{i,j}\langle v_i|v_j\rangle}$, and
2. $\frac{\sum_{i,j\in[n]} J_{i,j}\langle v_i|v_j\rangle}{\sum_{i,j\in[n]} J_{i,j}\langle s_i|s_j\rangle}$

It seems that in some cases it is easier to give these two bounds separately, rather than directly giving the bound between the vector (SDP) solution and the integer solution.

5.3 Product rules for copositive programming

In many applications (like parallel repetition), it is important to check whether a completely positive solution multiplies. Intuitively, given two copositive programs $\pi_1 = (J_1, \mathbf{A}_1, \mathbf{B}_1)$ and $\pi_2 = (J_2, \mathbf{A}_2, \mathbf{B}_2)$, can their solutions be multiplied to get a solution of $\pi_1 \otimes \pi_2$. Here $\pi_1 \otimes \pi_2$ needs to be defined, but mostly it is taken to be the copositive program $\pi_1 \otimes \pi_2 = (J_1 \otimes J_2, \mathbf{A}_1 \otimes \mathbf{A}_2, \mathbf{B}_1 \otimes \mathbf{B}_2)$. Formally, there are two notions of "multiplies" here.

1. Given any solutions x_i of π_1 and y_j of π_2, $x_i \otimes y_j$ is a solution of $\pi_1 \otimes \pi_2$.
2. Given optimal solutions x_i of π_1 and y_j of π_2, $x_i \otimes y_j$ is an "optimal" solution of $\pi_1 \otimes \pi_2$.

Note that π_1 and π_2 can be same here, and then we might ask the same question for higher powers too. The first condition is satisfied for all copositive programs of type 5.1.4, because of elementary properties of tensor products. But whether optimal solutions multiply to give optimal solutions is an open question.

5.4 Parallel repetition of unique games

We try to simplify the approach of Barak et al. ([BHH+08]), who prove that the value of any parallel repeated game approaches the value of the semidefinite relaxation of that game. The value of a nonlocal unique game can be upper bounded by the SDP (E denotes the expected value):

$$\begin{aligned} \min \quad & E_{u,u',\pi} \textstyle\sum_{i\in[k]} \|x_{u,i} - x_{u',\pi(i)}\|^2 \\ s.t. \quad & \textstyle\sum_{i\in[k]} ||x_{u,i}||^2 = 1 \\ & \langle x_{u,i} i | x_{u,j} \rangle = 0 \ \ \forall \ \ i \neq j \end{aligned} \tag{5.4.1}$$

We have the solution of this SDP, whose value is $1-\delta$. The goal is to construct a strategy of the parallely repeated game (repeated l times), whose value is $1 - O(\sqrt{l\delta \log k})$. The rounding algorithm by CMM ([CMM06]), gives us a way to get an integer solution with value $1-O(\sqrt{\delta \log k})$ (k is the alphabet size). Let us attempt to get the required value directly. If we multiply the rounded strategy (of CMM algorithm) l times, we get a solution of G^l with value $1 - O(l\sqrt{\delta \log k})$. Instead, if we multiply the SDP solution, we get an SDP solution for game G^l with value $1 - l\delta$. Then rounding the SDP solution of parallely repeated game using CMM algorithm ([CMM06]), we get a solution of G^l with value $1 - O(\sqrt{l\delta l \log k})$. This is because the alphabet size is k^l now. So in both the cases we get a value of $1 - O(l\sqrt{\delta \log k})$, but the required value is $1 - O(\sqrt{l\delta \log k})$.

Now we will use completely positive solution to get over this problem. We claim that we can generate completely positive solution from SDP solution with value $1-O(\delta \log \frac{k}{\delta})$ (Step 1). Since this solution multiplies, we can convert this to a completely positive solution of G^l with value $1 - O(l\delta \log \frac{k}{\delta})$ (Step 2). Then we convert this solution into solution of G^l with value $1 - O(\sqrt{l\delta \log \frac{k}{\delta}}$ (Step 3). Step 2 is clear because completely

positive solution multiplies as discussed in the previous section. Now we will present the proof of Step 1 and Step 3 formally.

The *distributional strategy* used by Boaz et.al. ([BHH+08]), is equivalent to a completely positive solution of this SDP (Equation 5.4.1). The concept of hellinger distance is equal to the objective value of the corresponding copositive program. They prove that

Theorem 5.4.1. *Say, the value of the SDP 5.4.1 for a nonlocal unique game is* $SDPval(G) = 1 - \delta$. *Then,*

$$val(G^l) \geq 1 - O(\sqrt{l\delta \log \frac{k}{\delta}}) \geq (1-\delta)^{O(l)}.$$

Proof. Consider the SDP mentioned in Equation 5.4.1. Let us assume that $x_{u,i} \in \mathcal{R}^d$ is the solution of this SDP, s.t., $SDPval(G) = 1 - \delta$. Let $y_{u,i}$ be the corresponding completely positive solution. For Step 1, we need to show that the objective value of $y_{u,i}$ is more than $1 - O(\delta \log \frac{k}{\delta})$. As before, this completely positive solution can be thought of as a solution of this copositive programming problem (not an SDP).

$$\begin{aligned} min \quad & E_{u,u',\pi} \textstyle\sum_{i\in[k]} ||y_{u,i} - y_{u',\pi(i)}||^2 \\ s.t. \quad & \textstyle\sum_{i\in[k]} ||y_{u,i}||^2 = 1 \\ & \langle y_{u,i} i | y_{u,j} \rangle = 0 \ \ \forall \ \ i \neq j \\ & y_{u,i} \geq 0 \end{aligned} \tag{5.4.2}$$

First we will define new vectors $y_{u,i,j}$ in terms of $x_{u,i}$. $y_{u,i,j}$ will have a coordinate for every vector in $\mathcal{R}^d$ (remember $x_{u,i} \in \mathcal{R}^d$). So every coordinate will be indexed by a vector $v \in \mathcal{R}^d$.

$$(y_{u,i,j})_v = \begin{cases} 0 & \text{If v's projection on } x_{u,i} \text{ is not maximum out of all } x_{u,h} \\ \sqrt{\gamma_{u,j,v}} & \text{otherwise} \end{cases} \tag{5.4.3}$$

Here $\gamma_{u,j,v}$ is the probability of selecting v with Gaussian centered on $x_{u,j}$. Then we define the v^{th} coordinate of the completely positive solution $y_{u,i}$ in terms of $y_{u,i,j}$, with

relation

$$(y_{u,i})_v^2 = \sum_{j\in[k]} \|x_{u,j}\|^2 (y_{u,i,j})_v^2 \tag{5.4.4}$$

So $y_{u,i}$ is also a vector of dimension $|\mathcal{R}^d|$ (remember, $x_{u,i} \in \mathcal{R}^d$). They can be explicitly defined by

$$(y_{u,i})_v = \begin{cases} 0 & \text{If v's projection on } x_{u,i} \text{ is not maximum out of all } x_{u,j} \\ \sqrt{\sum_{j\in[k]} ||x_{u,j}||^2 \gamma_{u,j,v}} & \text{otherwise} \end{cases} \tag{5.4.5}$$

It is clear that $y_{u,i}$ is orthogonal to $y_{u,j}$. Also, for all (u, j) pairs, the vectors $y_{u,i,j}$ (considering all i) define a probability distribution. Then, for all u, the vectors $y_{u,i}$ also define a probability distribution (they are convex combinations of $y_{u,i,j}$). So $y_{u,i}$ are the completely positive solutions of SDP 5.4.1. To simplify the notation, for any set of vectors $z_{u,i}$, lets call $obj(z_{u,i}) = E_{u,u',\pi} \sum_{i\in[k]} \|y_{u,i} - y_{u',\pi(i)}\|^2$.

To prove the Step 1, we need to show

$$obj(y_{u,i}) \geq 1 - O(\delta \log \frac{k}{\delta})$$

From Claim 5.2 of article [BHH+08], we can choose t (depending upon σ of Gaussian distribution), so that for every permutation π, we know

$$\sum_{i\in[k]} ||y_{u,i,j} - y_{u,\pi(i),\pi(j)}||^2 \geq (O(t)||x'_{u,i} - x'_{u',\pi i}||^2 + k.2^{-t})$$

Here $x'_{u,i}$ is the unit vector in the direction of $x_{u,i}$. Using this, we can prove that

$$\sum_{i\in[k]} ||y_{u,i} - y_{u,\pi(i)}||^2 \geq (O(t) \sum_{i\in[k]} \frac{1}{2}||x_{u,i} - x_{u',\pi i}||^2 + k.2^{-t})$$

Now choosing $t = \log \frac{k}{\delta}$, we get the required result

$$obj(y_{u,i}) \geq 1 - O(\delta \log \frac{k}{\delta})$$

Since this completely positive solution multiplies (Step 2), we know

$$obj(y_{u,i}^{\otimes l}) \geq 1 - O(l\delta \log \frac{k}{\delta}) \tag{5.4.6}$$

To convert these vectors into strategy of G^l (Step 3), we choose $v \in (\mathbb{R}^d)^{\otimes l}$ using correlated sampling lemma. Then answer $(i_1, i_2, \cdots, i_l)$, if $y_{u,i_1,i_2\cdots i_l}$'s v^{th} coordinate is non-zero. Using lemma 4.5 of article [BHH+08], we know

$$val(G^l) \geq 1 - O(\sqrt{l\delta \log \frac{k}{\delta}}) \tag{5.4.7}$$

Hence the result is proved.

□

Chapter 6

Conclusions

In this work, we have shown two applications of semidefinite programming. The first one was in obtaining product rules. Secondly we characterized quantum query complexity in terms of SDPs. The advantage of expressing (approximating) an optimization problem in terms of semidefinite program is that it can be solved efficiently. With the help of duality theory, other important properties of these optimization problems can be derived.

We now have a generalized framework for deriving product rules of quantities which can be estimated using SDPs. There are sufficient and necessary conditions under which two semidefinite instances obey the product rule. These conditions help us in determining, which constraints need to be relaxed or changed, in order to make the program "multiplicative". There are still many open questions in this field, and this study is by no means complete.

For the other application, we give a characterization of quantum query complexity in terms of an SDP. The dual formulation motivates the design of an algorithm, which computes the function value using the solution of the SDP. Hence, it proves that the adversary bound is tight for quantum query complexity. An equivalent SDP formulation shows the composition property of quantum query complexity in a simplified manner. This semidefinite formulation might also be useful in getting a strong direct product theorem.

There is hope that other convex optimization techniques can also be fruitful in complexity theory and quantum computing. We have explored one such application of copositive programming. It will be interesting to find other applications of these convex programming classes and their properties with respect to composition.

6.1 Open problems

There are still many question which need to be answered. Some of them are listed below.

1. Product rules

    - Can we give a unified sufficient and necessary condition for product rules to hold for semidefinite programs? We presently have a necessary condition that depends upon the dual solution. The goal is to remove that dependence, and to show that the obtained condition is sufficient also.
    - Since linear programs are a special case of semidefinite programs, is it easier to get the sufficient and necessary condition for them?

2. Quantum query complexity

    - Can we show a strong direct product theorem for quantum query complexity?
    - What is the quantum query complexity for functions having variable output size?

3. Other problems

    - Can a general framework for obtaining bounds on other compositions (other than multiplication) be developed for semidefinite programs?
    - Are there other applications of copositive programming?
    - What are the conditions under which a general convex program "multiplies"?

We hope that these questions will be solved in the near future.

References

[ACGT10] A. Ambainis, A. Childs, F. Le Gall, and S. Tani. The quantum query complexity of certification. *Quantum Inf. Comput.*, 10:181–188, 2010, arXiv:0903.1291 [quant-ph].

[ACR+10] A. Ambainis, A. Childs, B. Reichardt, R. Špalek, and S. Zhang. Any AND-OR formula of size N can be evaluated in time $N^{1/2+o(1)}$ on a quantum computer. *SIAM J. Comput.*, 39(6):2513–2530, 2010. Earlier version in FOCS'07.

[Ali95] F. Alizadeh. Interior Point Methods in Semidefinite Programming with Applications to Combinatorial Optimization In *SIAM J. Optim.*, 5(1):13–51, 1995.

[Amb02] A. Ambainis. Quantum lower bounds by quantum arguments. *J. Comput. Syst. Sci.*, 64:750–767, 2002, arXiv:quant-ph/0002066. Earlier version in STOC'00.

[Amb03] A. Ambainis. Polynomial degree vs. quantum query complexity. In *Proceedings of the 44th IEEE Symposium on Foundations of Computer Science*, 230–239. IEEE, 2003.

[Amb06] A. Ambainis. Polynomial degree vs. quantum query complexity. *J. Comput. Syst. Sci.*, 72(2):220–238, 2006, arXiv:quant-ph/0305028. Earlier version in FOCS'03.

[Amb07] A. Ambainis. Quantum walk algorithm for element distinctness. *SIAM J. Computing*, 37(1):210–239, 2007, arXiv:quant-ph/0311001. Earlier version in FOCS'04.

[ARV04] S. Arora, S. Rao, and U. Vazirani. Expander flows, geometric embeddings, and graph partitionings. In *Proceedings of the 36th ACM Symposium on the Theory of Computing*. ACM, 2004.

[Bha07] Rajendra Bhatia. *Positive Definite Matrices*. Princeton University Press, Princeton, 2007.

[BBBV97] Charles H. Bennett, Ethan Bernstein, Gilles Brassard, and Umesh Vazirani. Strengths and weaknesses of quantum computing. *SIAM J. Comput.*, 26(5):1510–1523, 1997, arXiv:quant-ph/9701001.

[BDH+00] Harry Buhrman, Christoph Dürr, Mark Heiligman, Peter Høyer, Frédéric Magniez, Miklos Santha, and Ronald de Wolf. Quantum algorithms for element distinctness. 2000, arXiv:quant-ph/0007016.

[BFL91] L. Babai, L. Fortnow, and C. Lund. Non-deterministic exponential time has two-prover interactive protocols. *Computational Complexity*, 1:3–40, 1991.

[BHH+08] B. Barak, M. Hardt, I. Haviv, A. Rao, O. Regev, and D. Steurer. Rounding parallel repetitions of unique games. In *Proceedings of the 49th IEEE Symposium on Foundations of Computer Science*. IEEE, 2008.

[BK00] H. Barnum and E. Knill. Reversing quantum dynamics with near-optimal quantum and classical fidelity. J. Math. Phys. 43:2097, 2002.

[BNRW05] Harry Buhrman, Ilan Newman, Hein Röhrig, and Ronald de Wolf. Robust polynomials and quantum algorithms. In *Proc. 22nd STACS*, LNCS vol. 3404:593–604, 2005, arXiv:quant-ph/0309220.

[BOH07] Michael Ben-Or and Avinatan Hassidim. Quantum search in an ordered list via adaptive learning. 2007, arXiv:quant-ph/0703231.

[BS04] Howard Barnum and Michael Saks. A lower bound on the quantum query complexity of read-once functions. *J. Comput. Syst. Sci.*, 69(2):244–258, 2004, arXiv:quant-ph/0201007.

[BSS03] H. Barnum, M. Saks, and M. Szegedy. Quantum query complexity and semidefinite programming. In *Proceedings of the 18th IEEE Conference on Computational Complexity*, 179–193, 2003.

[BV04] S. Boyd and L. Vandenberghe. *Convex optimization*. Cambridge University Press, 2004.

[CCJY09] Andrew M. Childs, Richard Cleve, Stephen P. Jordan, and David Yeung. Discrete-query quantum algorithm for NAND trees. *Theory of Computing*, 5:119–123, 2009, arXiv:quant-ph/0702160.

[CE05] Andrew M. Childs and Jason M. Eisenberg. Quantum algorithms for subset finding. *Quantum Inf. Comput.*, 5(7):593–604, 2005, arXiv:quant-ph/0311038.

[CHTW04] R. Cleve, P. Høyer, B. Toner, and J. Watrous. Consequences and limits of nonlocal strategies. In *Proceedings of the 19th IEEE Conference on Computational Complexity*, 236–249. IEEE, 2004.

[CL08] Andrew M. Childs and Troy Lee. Optimal quantum adversary lower bounds for ordered search. In *Proc. 35th ICALP*, LNCS vol. 5125:869–880, 2008, arXiv:0708.3396 [quant-ph].

[CLP07] Andrew M. Childs, Andrew J. Landahl, and Pablo A. Parrilo. Improved quantum algorithms for the ordered search problem via semidefinite programming. *Phys. Rev. A*, 75:032335, 2007, arXiv:quant-ph/0608161.

[CMM06] M. Charikar, K. Makarychev, Y. Makarychev. Near-optimal algorithms for unique games. In *Proceedings of 38th IEEE Symposium on Foundations of Computer Science* , pages 205-214.ACM, 2006.

[CSUU07] R. Cleve, W. Slofstra, F. Unger, and S. Upadhyay. Perfect parallel repetition theorem for quantum XOR proof systems. In *Proceedings of the 22nd IEEE Conference on Computational Complexity*. IEEE, 2007.

[FGG08] Edward Farhi, Jeffrey Goldstone, and Sam Gutmann. A quantum algorithm for the Hamiltonian NAND tree. *Theory of Computing*, 4:169–190, 2008, arXiv:quant-ph/0702144.

[FGGS99] Edward Farhi, Jeffrey Goldstone, Sam Gutmann, and Michael Sipser. Invariant quantum algorithms for insertion into an ordered list. 1999, arXiv:quant-ph/9901059.

[FL92] U. Feige and L. Lovász. Two-prover one-round proof systems: their power and their problems. In *Proceedings of the 24th ACM Symposium on the Theory of Computing*, pages 733–744. ACM, 1992.

[GW95] M. Goemans and D. Williamson. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *Journal of the ACM*, 42:1115–1145, 1995.

[Hol73] A. S. Holevo. *Statistical decision theory for quantum systems.* J. Multivariate Anal, 3:337, 1973.

[HJ85] R. Horn and C. Johnson. *Matrix Analysis.* Chapter 7, Cambridge University Press, 1985.

[HLŠ07] P. Høyer, T. Lee, and R. Špalek. Negative weights make adversaries stronger. In *Proceedings of the 39th ACM Symposium on the Theory of Computing*. ACM, 2007.

[HNS02] Peter Høyer, Jan Neerbek, and Yaoyun Shi. Quantum complexities of ordered searching, sorting, and element distinctness. *Algorithmica*, 34(4):429–448, 2002, arXiv:quant-ph/0102078. Special issue on Quantum Computation and Cryptography.

[HŠ05] Peter Høyer and Robert Špalek. Lower bounds on quantum query complexity. *EATCS Bulletin*, 87:78–103, October 2005, arXiv:quant-ph/0509153.

[Hol07] T. Holenstein. Parallel repetition theorem: simplifications and the no-signaling case. In *Proceedings of the 39th ACM Symposium on the Theory of Computing*, 411–419, 2007.

[JJUW10] R. Jain, Z. Ji, S. Upadhyay, J. Watrous. QIP=PSPACE. In *Proceedings of the 42nd ACM Symposium on Theory of Computing*, 2010.

[JKN08] R. Jain, H. Klauck, A. Nayak. Direct product theorems for classical communication complexity via subdistribution bounds. In *Proceedings of the 40th ACM Symposium on the Theory of Computing*, 599-608, 2008.

[JKS10] Rahul Jain, Hartmut Klauck, and M. Santha. Optimal direct sum results for deterministic and randomized decision tree complexity. 2010, arXiv:1004.0105 [cs.CC].

[Kar84] N. Karmarkar. A new polynomial-time algorithm for linear programming. In *Combinatorica* 4:373-395, 1984.

[KKMO07] S. Khot, G. Kindler, E. Mossel, and R. O'Donnell. Optimal inapproximability results for MAX-CUT and other two-variable CSPs?. *SIAM Journal on Computing*, 37(1):319-357, 2007.

[KKN95] M. Karchmer, E. Kushilevitz, and N. Nisan. Fractional covers and communication complexity. *SIAM Journal on Discrete Mathematics*, 8(1):76–92, 1995.

[KMS98] D. Karger, R. Motwani, and M. Sudan. Approximate graph coloring by semidefinite programming. *Journal of the ACM*, 45(2):246–265, 1998.

[Knu94] D. Knuth. The sandwich theorem. *Electronic Journal of Combinatorics*, volume 1, article A1, 1994.

[KRT07] J. Kempe, O. Regev, and B. Toner. The unique game conjecture with entangled provers is false. Technical Report 0712.4279, arXiv, 2007.

[KSV02] Alexei Yu. Kitaev, Alexander H. Shen, and Mikhail N. Vyalyi. *Classical and Quantum Computation*, volume 47 of *Graduate Studies in Mathematics*. American Mathematical Society, Providence, Rhode Island, 2002.

[KN97] E. Kushilevitz and N. Nisan. *Communication Complexity*. Cambridge University Press, 1997.

[KR10] J. Kempe and O. Regev. *No Strong Parallel Repetition with Entangled and Non-signaling Provers*. Conference on Computational Complexity, 2010.

[KW00] A. Kitaev and J. Watrous. Parallelization, amplification, and exponential time simulation of quantum interactive proof systems. In *Proceedings of the 32nd ACM Symposium on Theory of Computing*, 608617, 2000.

[LLS06] S. Laplante, T. Lee, and M. Szegedy. The quantum adversary method and classical formula size lower bounds. *Computational Complexity*, 15:163–196, 2006.

[LM04] Sophie Laplante and Frédéric Magniez. Lower bounds for randomized and quantum query complexity using Kolmogorov arguments. In *Proc. 19th IEEE Complexity*, 294–304, 2004, arXiv:quant-ph/0311189.

[LM08] T. Lee and R. Mittal. Product theorems via semidefinite programming. In *Proceedings of the 35th International Colloquium On Automata, Languages and Programming*, volume 5125 of *Lecture Notes in Computer Science*, 674–685. Springer-Verlag, 2008. arXiv:0803.4206.

[LMRS10] T. Lee, R. Mittal, B. Reichardt and R. Špalek. An adversary for algorithms. arXiv:1011.3020v1.

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

[Lov] L. Lovász. Semidefinite programs and combinatorial optimization, Lecture Notes. *http://www.cs.elte.hu/lovasz/semidef.ps.*

[Lov75] L. Lovász. On the ratio of optimal integral and fractional covers. *Discrete Mathematics*, 13:383–390, 1975.

[Lov79] L. Lovász. On the Shannon capacity of a graph. *IEEE Transactions on Information Theory*, IT-25:1–7, 1979.

[LS08] N. Linial and A. Shraibman. Learning complexity versus communication complexity. In *Proceedings of the 23rd IEEE Conference on Computational Complexity*. IEEE, 2008.

[LSŠ08] T. Lee, A. Shraibman, and R. Špalek. A direct product theorem for discrepancy. In *Proceedings of the 23rd IEEE Conference on Computational Complexity*, 71–80. IEEE, 2008.

[Moc07] C. Mochon. Quantum weak coin flipping with arbitrarily small bias. Technical Report arXiv:0711.4114, arXiv, 2007.

[MS07] R. Mittal and M. Szegedy. Product rules in semidefinite programming. In *16th International Symposium on Fundamentals of Computation Theory*, 435–445. Springer, 2007.

[MSS05] Frédéric Magniez, Miklos Santha, and Mario Szegedy. Quantum algorithms for the triangle problem. In *Proc. 16th ACM-SIAM Symp. on Discrete Algorithms (SODA)*, 2005, arXiv:quant-ph/0310134.

[Rag08] P. Raghavendra. Optimal algorithms and inapproximability results for every CSP? *Proceedings of the 40th ACM Symposium on the Theory of Computing*, 245–254, 2008.

[Raz98] R. Raz. A parallel repetition theorem. *SIAM Journal on Computing*, 27(3):763–803, 1998.

[Rei09] Ben W. Reichardt. Span programs and quantum query complexity: The general adversary bound is nearly tight for every boolean function. 2009, arXiv:0904.2759 [quant-ph]. Extended abstract in *Proc. 50th IEEE FOCS, 544–551, 2009.*

[Rei10a] *Ben W. Reichardt. Reflections for quantum query algorithms. 2010,* arXiv:1005.1601 [quant-ph].

[Rei10b] *Ben W. Reichardt. Least span program witness size equals the general adversary lower bound on quantum query complexity. Technical Report TR10-075, Electronic Colloquium on Computational Complexity,* http://eccc.hpi-web.de*, 2010.*

[Rei10c] *Ben W. Reichardt. Span programs and quantum query algorithms. Technical Report TR10-110, Electronic Colloquium on Computational Complexity,* http://eccc.hpi-web.de*, 2010.*

[RŠ08] *Ben W. Reichardt and Robert Špalek. Span-program-based quantum algorithm for evaluating formulas. In* Proc. 40th ACM STOC*, 103–112, 2008,* arXiv:0710.2630 [quant-ph]*.*

[Sha03] *R. Shaltiel. Towards proving strong direct product theorems.* Computational Complexity*, 12(1–2):1–22, 2003.*

[Shi02] *Y. Shi. Quantum lower bounds for the collision and the element distinctness problems. In* Proc. 43rd IEEE FOCS*, 513–519, 2002,* arXiv:quant-ph/0112086*.*

[Sze94] *M. Szegedy. A note on the theta number of Lovász and the generalized Delsarte bound. In* Proceedings of the 35th IEEE Symposium on Foundations of Computer Science*, 36–39. IEEE, 1994.*

[ŠS06] *R. Špalek and M. Szegedy. All quantum adversary methods are equivalent.* Theory of Computing*, 2(1):1–18, 2006,* arXiv:quant-ph/0409116*. Earlier version in ICALP'05.*

[Tsi87] *B. Tsirelson. Quantum analogues of the Bell inequalities: the case of two spatially separated domains.* Journal of Soviet Mathematics*, 36:557–570, 1987.*

[VB96] *L. Vandenberghe and S. Boyd. Semidefinite programming.* SIAM Review*, 38:49–95, 1996.*

[YKL75] *H. Yuen, R. Kennedy and M. Lax. Optimum testing of multiple hypotheses in quantum detection theory. In* IEEE Transactions on Information Theory*, 21(2):125–134, 1975.*

[Zha05] *Shengyu Zhang. On the power of Ambainis's lower bounds.* Theoretical Computer Science*, 339(2-3):241–256, 2005,* arXiv:quant-ph/0311060*. Earlier version in ICALP'04.*

Appendix A

Bipartite tensor product

This definition and argument is taken from [KRT07]. In the context of non local classical and quantum games, another definition of product of semidefinite instances is interesting. Suppose for the semidefinite instance $(J, \mathbf{A}, b)$, it is given that J is anti block diagonal and $\mathbf{A}$ is block diagonal. So

$$J \qquad\qquad A^{(i)}$$

$$\begin{pmatrix} 0 & K \\ K^T & 0 \end{pmatrix} \qquad \begin{pmatrix} P^{(i)} & 0 \\ 0 & Q^{(i)} \end{pmatrix}$$

Then the semidefinite instances $\pi_c(J, \mathbf{A}, b)$, $c \in \{0, 1\}$, can be described as $K_c, P_c^{(i)}, Q_c^{(i)}, b_c$. We can now define the "bipartite tensor product" between π_1 and π_2 as $\pi_1 \otimes_b \pi_2$.

$$J \qquad\qquad A^{(ij)} \qquad\qquad b$$

$$\begin{pmatrix} 0 & K_1 \otimes K_2 \\ K_1^T \otimes K_2^T & 0 \end{pmatrix} \qquad \begin{pmatrix} P_1^{(i)} \otimes P_2^{(j)} & 0 \\ 0 & Q_1^{(i)} \otimes Q_2^{(j)} \end{pmatrix} \qquad b_1 \otimes b_2$$

Here, we see that the dimensions of the variable matrix is smaller compared to the original tensor product. Now Theorem 3.3.4 proves that dual is feasible for the original tensor product instance.

$$(y_1 \otimes y_2)^T (\mathbf{A}_1 \otimes \mathbf{A}_2) - J_1 \otimes J_2 \succeq 0 \tag{A.0.1}$$

To show that product theorem also holds for $\pi_1 \otimes_b \pi_2$, we need to show that $y_1 \otimes y_2$ is feasible for the dual of $\pi_1 \otimes \pi_2$. Hence, we need to show that the following matrix is positive semidefinite.

$$\begin{pmatrix} \sum_{ij}(y_1)_i(y_2)_j A_1^{(i)} \otimes A_2^{(j)} & 0 \\ 0 & \sum_{ij}(y_1)_i(y_2)_j B_1^{(i)} \otimes B_2^{(j)} \end{pmatrix} - \begin{pmatrix} 0 & K_1 \otimes K_2 \\ K_1^T \otimes K_2^T & 0 \end{pmatrix}$$

Notice that this is just an independent sub block of the matrix in Equation A.0.1. Since the complete matrix in Eq. A.0.1 is positive semidefinite, implies this matrix is also positive semidefinite. Hence, product theorem follows for "bipartite tensor product" also.

Appendix B

Quantum query complexity

B.1 Proof of Proposition 4.4.1

Proof of Lemma 4.4.2. By definition of $B_{G(x)}$, we have $B_{G(x)}|\psi\rangle = (B_G|\psi\rangle, \overline{\Pi}(x)|\psi\rangle)$. The first term is

$$\begin{aligned} B_G|\psi\rangle &= \Big(|c\rangle\langle\varnothing| + \sum_{b\in E}|c_b\rangle\langle b| + \sum_{y\in\mathcal{D},j\in[n]}|y\rangle\langle j|\otimes\langle\mu_{y_j}|\otimes\big(\langle v_{yj}| + \langle w_{yj}|\big)\Big)|\psi\rangle \\ &= |c\rangle + |c_{f(x)}\rangle + \frac{\eta}{\sqrt{W}}\frac{2(k-1)}{k}\sum_{y\in\mathcal{D},j\in[n]}|y\rangle\langle\mu_{y_j}|\nu_{x_j}\rangle\big(\langle v_{yj}|v_{xj}\rangle - \langle w_{yj}|w_{xj}\rangle\big) \\ &= |c\rangle + |c_{f(x)}\rangle + \frac{\eta}{\sqrt{W}}\sum_{y\in\mathcal{D}}\sum_{j\in[n]:y_j\neq x_j}|y\rangle\big(\langle v_{yj}|v_{xj}\rangle - \langle w_{yj}|w_{xj}\rangle\big) \\ &= 0\ . \end{aligned} \tag{B.1.1}$$

Where in the third equation we used $\langle\mu_{y_j}|\nu_{x_j}\rangle$ is 0, if $x_j = y_j$, and is otherwise $k/(2(k-1))$. In the last step we used the SDP constraint $\sum_{j:x_j\neq y_j}\big(\langle v_{yj}|v_{xj}\rangle - \langle w_{yj}|w_{xj}\rangle\big) = 1 - \delta_{f(x),f(y)}$ and the definition of $|c_{f(x)}\rangle$. The second term, $\overline{\Pi}(x)|\psi\rangle$, evaluates to zero since $\langle\mu_{x_j}|\nu_{x_j}\rangle = 0$:

$$\begin{aligned} \overline{\Pi}(x)|\psi\rangle &= \frac{\eta}{\sqrt{W}}\overline{\Pi}(x)\sum_{j\in[n]}|j\rangle\otimes|\nu_{x_j}\rangle\otimes\big(|v_{xj}\rangle - |w_{xj}\rangle\big) \\ &= \frac{\eta}{\sqrt{W}}\sum_{j\in[n]}\langle\mu_{x_j}|\nu_{x_j}\rangle|j\rangle\otimes|\mu_{x_j}\rangle\otimes\big(|v_{xj}\rangle - |w_{xj}\rangle\big) = 0\ . \end{aligned} \tag{B.1.2}$$

Thus indeed $B_{G(x)}|\psi\rangle = 0$. The claim $|\langle\phi_+|\psi\rangle|^2/\||\psi\rangle\|^2 \geq \frac{2}{2+4\eta^2} > 1 - 2\eta^2$ is a calculation, using $\||\nu_{x_j}\rangle\| = 1$ and $\sum_j\big(\||v_{xj}\rangle\|^2 + \||w_{xj}\rangle\|^2\big) \leq W$.

For the second part of the lemma, we claim that $B^\dagger_{G'(x)}|\psi'\rangle = 0$. Indeed,

$$B^\dagger_{G'(x)}|\psi'\rangle = (\mathbf{1} - |\phi_-\rangle\langle\phi_-|)\Big(B^\dagger_G \quad \overline{\Pi}(x)\Big)\begin{pmatrix} -\frac{\eta}{\sqrt{W}}|c\rangle + |x\rangle \\ -\sum_j |j\rangle\otimes|\mu_{x_j}\rangle\otimes\big(|v_{xj}\rangle + |w_{xj}\rangle\big)\end{pmatrix}$$
$$= (\mathbf{1} - |\phi_-\rangle\langle\phi_-|)\begin{pmatrix} \big(|\varnothing\rangle\langle c| + \sum_{b\in E}|b\rangle\langle c_b|\big)\big(\frac{-\eta}{\sqrt{W}}|c\rangle + |x\rangle\big) \\ +A^\dagger|x\rangle - \overline{\Pi}(x)\sum_j |j\rangle\otimes|\mu_{x_j}\rangle\otimes\big(|v_{xj}\rangle + |w_{xj}\rangle\big)\end{pmatrix} \tag{B.1.3}$$

The term $A^\dagger|x\rangle$ cancels the $\overline{\Pi}(x)$ term, while the other terms are proportional to $|\phi_-\rangle$:

$$\big(|\varnothing\rangle\langle c| + \sum_{b\in E}|b\rangle\langle c_b|\big)\big(\frac{-\eta}{\sqrt{W}}|c\rangle + |x\rangle\big) = \Big(\frac{-\eta}{\sqrt{W}}\big(|\varnothing\rangle - \sum_{b\in E}|b\rangle\big) - \frac{\eta}{\sqrt{W}}\sum_{b:f(x)\neq b}|b\rangle\Big)$$
$$= \frac{-\eta}{\sqrt{W}}\big(|\varnothing\rangle - |f(x)\rangle\big) = \frac{-\eta}{\sqrt{W}}\sqrt{2}|\phi_-\rangle \ . \tag{B.1.4}$$

Finally, use $B_{G(x)}|\phi_-\rangle = \frac{1}{\sqrt{2}}\big(2|c\rangle + \frac{\eta}{\sqrt{W}}\sum_{y\in\mathcal{D}:f(y)\neq f(x)}|y\rangle\big)$ to calculate $|\langle\phi_-|B^\dagger_{G(x)}|\psi'\rangle|^2/\||\psi'\rangle\|^2 \geq \frac{2\eta^2}{W}/(\frac{\eta^2}{W} + 1 + W) > 9/(10^5W^2)$. □

Proof of Proposition 4.4.1. :

Let $|\varphi\rangle = \sum_{\beta:\theta(\beta)=0}|\beta\rangle\langle\beta|\varnothing\rangle$. By Lemma 4.4.4 with $\Theta = 0$, $|\varphi\rangle = \frac{1}{\sqrt{2}}\sum_{\beta:\theta(\beta)=0}|\beta\rangle\langle\beta|\phi_+\rangle$ and $\||\varphi\rangle\|^2 > \frac{1}{2}(1 - 2\cdot 10^{-4})$. Since $U_x|\varphi\rangle = |\varphi\rangle$, either $\Delta|\varphi\rangle = \Pi_x|\varphi\rangle = |\varphi\rangle$ or $\Delta|\varphi\rangle = \Pi_x|\varphi\rangle = -|\varphi\rangle$. The former case holds as $\Pi_x|\phi_+\rangle = |\phi_+\rangle$. Thus $\Delta|\varphi\rangle = |\varphi\rangle$, so $\langle f(x)|\varphi\rangle = \langle\varnothing|\varphi\rangle$. This then implies $\||\varphi\rangle\|^2 = \frac{1}{2}\langle\phi_+|\sum_{\beta:\theta(\beta)=0}|\beta\rangle\langle\beta|\phi_+\rangle = \frac{1}{\sqrt{2}}\langle\phi_+|\varphi\rangle = \frac{1}{2}(\langle\varnothing|\varphi\rangle + \langle f(x)|\varphi\rangle) = \langle f(x)|\varphi\rangle$. (Let us remark that $|\varphi\rangle$ need not be proportional to the state $|\psi\rangle$ from Eq. (4.4.9), and can have small overlap on vertices $|b\rangle$ for $b \in E\setminus\{f(x)\}$.)

Now let $\Theta = 3\cdot 10^{-7}/W$. Using notation from Lemma 4.4.4, define the projections $\Delta_\Theta = \sum_{\beta:0<|\theta(\beta)|\leq\Theta}|\beta\rangle\langle\beta|$ and $\overline{\Delta}_\Theta = \sum_{\beta:|\theta(\beta)|>\Theta}|\beta\rangle\langle\beta|$, so $|\varnothing\rangle = |\varphi\rangle + \Delta_\Theta|\varnothing\rangle + \overline{\Delta}_\Theta|\varnothing\rangle$. By Eq. (4.4.17), $\|\Delta_\Theta|\phi_+\rangle\|^2 < 2\cdot 10^{-4}$. By Eq. (4.4.18), $\|\Delta_\Theta|\phi_-\rangle\|^2 < 2\cdot 10^{-4}$. Therefore, since $|\varnothing\rangle = \frac{1}{\sqrt{2}}\big(|\phi_+\rangle + |\phi_-\rangle\big)$,

$$\|\Delta_\Theta|\varnothing\rangle\| \leq \frac{1}{\sqrt{2}}\big(\|\Delta_\Theta|\phi_+\rangle\| + \|\Delta_\Theta|\phi_-\rangle\|\big) < \frac{1}{50} \ . \tag{B.1.5}$$

The algorithm outputs $b \in E$ with probability

$$\Big|\Big(\frac{1}{\sqrt{\tau}}\sum_{t\in[\tau]}\langle t|\otimes\langle b|\Big)\Big(\frac{1}{\sqrt{\tau}}\sum_{t\in[\tau]}|t\rangle\otimes U_x^t|\varnothing\rangle\Big)\Big|^2 = \frac{1}{\tau^2}\Big|\sum_{t\in[\tau]}\langle b|U_x^t|\varnothing\rangle\Big|^2 \ . \tag{B.1.6}$$

Then

$$\begin{aligned}\frac{1}{\tau}\Big|\sum_{t\in[\tau]}\langle f(x)|U_x^t|\phi\rangle\Big| &= \frac{1}{\tau}\Big|\sum_{t\in[\tau]}\langle f(x)|U_x^t\big(|\varphi\rangle+\Delta_\Theta|\phi\rangle+\overline{\Delta}_\Theta|\phi\rangle\big)\Big| \\ &\geq \langle f(x)|\varphi\rangle - \|\Delta_\Theta|\phi\rangle\| - \frac{1}{\tau}\Big|\sum_{t\in[\tau]}\langle f(x)|U_x^t\overline{\Delta}_\Theta|\phi\rangle\Big| \\ &> \frac{1}{2}(1-2\cdot 10^{-4}) - \frac{1}{50} - \frac{1}{\tau}\Big|\sum_{t\in[\tau],\beta:|\theta(\beta)|>\Theta}\langle f(x)|U_x^t|\beta\rangle\langle\beta|\phi\rangle\Big| \\ &= 0.48 - 10^{-4} - \frac{1}{\tau}\Big|\sum_{\beta:|\theta(\beta)|>\Theta}\frac{e^{i\theta(\beta)\tau}-1}{e^{i\theta(\beta)}-1}e^{i\theta(\beta)}\langle f(x)|\beta\rangle\langle\beta|\phi\rangle\Big| \ .\end{aligned} \tag{B.1.7}$$

Using $|e^{i\Theta}-1| = 2\sin\frac{\Theta}{2}$, the final term is at most

$$\frac{1}{\tau\sin\frac{\Theta}{2}}\sum_\beta|\langle f(x)|\beta\rangle\langle\beta|\phi\rangle| \leq \frac{1}{\tau\sin\frac{\Theta}{2}} < 0.01 \ , \tag{B.1.8}$$

by the Cauchy-Schwarz inequality. Therefore, the algorithm indeed outputs $f(x)$ with probability at least $0.46^2 > 21\%$.

Similarly, we can argue that for $b \in E \setminus \{f(x)\}$,

$$\begin{aligned}\frac{1}{\tau}\Big|\sum_{t\in[\tau]}\langle b|U_x^t|\phi\rangle\Big| &\leq |\langle b|\varphi\rangle| + \|\Delta_\Theta|\phi\rangle\| + \frac{1}{\tau}\Big|\sum_{t\in[\tau]}\langle b|U_x^t\overline{\Delta}_\Theta|\phi\rangle\Big| \\ &< |\langle b|\varphi\rangle| + \frac{1}{50} + 0.01 \ .\end{aligned} \tag{B.1.9}$$

Bound $|\langle b|\varphi\rangle|^2 \leq \||\varphi\rangle\|^2 - |\langle\phi|\varphi\rangle|^2 - |\langle f(x)|\varphi\rangle|^2 = \||\varphi\rangle\|^2 - 2\||\varphi\rangle\|^4 < \frac{1}{2} - 2\cdot\frac{1}{4}(1-2\cdot 10^{-4})^2 < 2\cdot 10^{-4}$. Thus the algorithm outputs b with probability at most $2\cdot 10^{-3}$, as claimed. □

B.2 Proofs of composition results

In this section we give proofs of Lemma 4.5.1 and Claim 4.5.4.

Proof of Lemma 4.5.1. Let $M = \sum_{i\in[n]} m_i$. For an input $x \in C^M$, let $\tilde{x} = \vec{g}(x)$.

Let us begin with Eq. (4.5.3). Say that $\{v_{\tilde{x},i}, w_{\tilde{x},i}\}$ are optimal vectors for f. In other words, these vectors are a factorization of optimal matrices from Eq. (4.2.5), the

dual formulation of witness size, and satisfy

$$\begin{aligned} F_f[\tilde{x},\tilde{y}] = 1 - \delta_{f(\tilde{x}),f(\tilde{y})} &= \sum_{i:\tilde{x}_i\neq\tilde{y}_i} \langle v_{\tilde{x},i}|v_{\tilde{y},i}\rangle - \langle w_{\tilde{x},i}|w_{\tilde{y},i}\rangle \\ \mathrm{wsize}_s(f) &= \max_{\tilde{x}} \sum_i s_i\big(\|v_{\tilde{x},i}\|^2 + \|w_{\tilde{x},i}\|^2\big) \ . \end{aligned} \tag{B.2.1}$$

Similarly, say that $\{t_{x^i,j}, u_{x^i,j}\}$ are optimal vectors for g_i. Thus, by assumption we have

$$\begin{aligned} F_{f\circ\vec{g}}[x,y] &= \sum_{i:\tilde{x}_i\neq\tilde{y}_i} \big(\langle v_{\tilde{x},i}|v_{\tilde{y},i}\rangle - \langle w_{\tilde{x},i}|w_{\tilde{y},i}\rangle\big) \\ &= \sum_i \big(\langle v_{\tilde{x},i}|v_{\tilde{y},i}\rangle - \langle w_{\tilde{x},i}|w_{\tilde{y},i}\rangle\big)\left(\sum_{j:x^i_j\neq y^i_j} \langle t_{x^i,j}|t_{y^i,j}\rangle - \langle u_{x^i,j}|u_{y^i,j}\rangle\right) \\ &= \sum_{i,j:x^i_j\neq y^i_j} \begin{pmatrix} \langle v_{\tilde{x},i}|v_{\tilde{y},i}\rangle\langle t_{x^i,j}|t_{y^i,j}\rangle + \langle w_{\tilde{x},i}|w_{\tilde{y},i}\rangle\langle u_{x^i,j}|u_{y^i,j}\rangle \\ -\langle v_{\tilde{x},i}|v_{\tilde{y},i}\rangle\langle u_{x^i,j}|u_{y^i,j}\rangle - \langle w_{\tilde{x},i}|w_{\tilde{y},i}\rangle\langle t_{x^i,j}|t_{y^i,j}\rangle \end{pmatrix} . \end{aligned} \tag{B.2.2}$$

The key point here is that the witness size (or adversary) SDP for $f\circ\vec{g}$ imposes a constraint on the summation over all i,j where $x^i_j \neq y^i_j$. This includes those i,j such that $x^i_j \neq y^i_j$, yet $g_i(x^i) = g_i(y^i)$. With the adversary bound we have no control over the inner products in this case; the witness size SDP, on the other hand, allows us to pass to a summation over i where $\tilde{x}_i \neq \tilde{y}_i$ to a summation over all i in the second line above.

At this stage, we can read off the proper construction of vectors for $f\circ\vec{g}$. Define the "positive vectors" as the concatenation $(v_{\tilde{x},i}\otimes t_{x^i,j}, w_{\tilde{x},i}\otimes u_{x^i,j})$ and define the "negative vectors" as $(v_{\tilde{x},i}\otimes u_{x^i,j}, w_{\tilde{x},i}\otimes t_{x^i,j})$. This solution gives an objective value of

$$\begin{aligned} &\max_x \sum_{i,j} \begin{pmatrix} \|v_{\tilde{x},i}\|^2\|t_{x^i,j}\|^2 + \|w_{\tilde{x},i}\|^2\|u_{x^i,j}\|^2 \\ +\|v_{\tilde{x},i}\|^2\|u_{x^i,j}\|^2 + \|w_{\tilde{x},i}\|^2\|t_{x^i,j}\|^2 \end{pmatrix} \qquad \text{(B.2.3)} \\ &= \max_x \sum_i (\|v_{\tilde{x},i}\|^2 + \|w_{\tilde{x},i}\|^2) \sum_j (\|t_{x^i,j}\|^2 + \|u_{x^i,j}\|^2) \\ &\leq \mathrm{wsize}_s(f) \ , \qquad \text{(B.2.4)} \end{aligned}$$

which completes the proof of Eq. (4.5.3).

By Eq. (4.2.9), this also implies Eq. (4.5.2), except with a lost factor of two on the right-hand side. That this factor of two is unnecessary can be seen easily by repeating the above arguments, except beginning with an optimal vector solution to the general

adversary bound dual SDP Eq. (4.2.3) for f, and then considering only inputs x, y with $(f \circ \vec{g})(x) \neq (f \circ \vec{g})(y)$. □

Proof of Claim 4.5.4. Say that the matrix B_i has dimensions m_i-by-m_i', and let $M = \prod_i (m_i + m_i')$. We define $2^k M$ many eigenvectors of $A \circ_b (\bigotimes B_i)$ and show that they span the entire space. Then we show that they all have eigenvalues whose magnitude is bounded by the expression in the claim, and that at least one achieves this bound.

Consider the matrix

$$\bar{B}_i = \begin{pmatrix} 0 & B_i^{0,1} \\ (B_i^{0,1})^\dagger & 0 \end{pmatrix} \ . \tag{B.2.5}$$

Let $u_i^0 \oplus u_i^1$ be an eigenvector of this matrix with eigenvalue λ_i. Notice that this vector has the property that $B_i^{0,1} u_i^1 = \lambda_i u_i^0$ and $(B_i^{0,1})^\dagger u_i^0 = \lambda_i u_i^1$.

For $\lambda = (\lambda_1, \ldots, \lambda_k)$ a sequence of eigenvalues of $\bar{B}_1, \ldots, \bar{B}_k$ respectively, define a 2^k-by-2^k matrix A_λ whose (x, y) entry is given by $A_\lambda[x, y] = A[x, y] \prod_i \lambda_i^{[x_i \neq y_i]}$. Here $\lambda_i^{[1]} = \lambda_i$ and $\lambda_i^{[0]} = \|B_i^{0,1}\|$.

Let α be an eigenvector of A_λ with eigenvalue c. Notice that this means for all x,

$$\sum_y A[x, y] \alpha[y] \prod_i \lambda_i^{[x_i \neq y_i]} = c\, \alpha[x] \ . \tag{B.2.6}$$

Now we claim that $\gamma = \bigoplus_x \alpha[x] \otimes u_i^{x_i}$ is an eigenvector of $A \circ_b (\bigotimes B_i)$. For any x we have

$$\begin{aligned} \sum_y A[x, y] \alpha[y] \bigotimes B_i^{x_i, y_i} \left(\bigotimes u_i^{y_i} \right) &= \bigotimes u_i^{x_i} \left(\sum_y A[x, y] \alpha[y] \prod_i \lambda_i^{[x_i \neq y_i]} \right) \\ &= c\, \alpha[x] \bigotimes u_i^{x_i}. \end{aligned} \tag{B.2.7}$$

The first line holds because

- If $x_i = y_i$, then $B_i^{x_i, y_i} u_i^{y_i} = \|B_i^{0,1}\| u_i^{x_i}$.
- If $x_i \neq y_i$, then $B_i^{x_i, y_i} u_i^{y_i} = \lambda_i u_i^{x_i}$.

For every λ and c we have now defined an eigenvector. It can be verified that these eigenvectors span the entire space. Thus there can be no others as the eigenvectors of a symmetric matrix are orthogonal.

We now bound the eigenvalues of each of these eigenvectors. To do this it suffices to bound the spectral norm of the matrix

$$A_\lambda = A \circ \bigotimes_i \begin{pmatrix} \|B^{1,0}\| & \lambda_i \\ \lambda_i & \|B^{1,0}\| \end{pmatrix} \ . \tag{B.2.8}$$

Note that for any rank-one sign matrix C we have that $\|A \circ C\| = \|A\|$. A similar situation is at work here. Define a variable matrix to be one whose entries are variables or products of variables. For a m-by-n variable matrix B, let $B(a)$ be the matrix obtained by the assignment $x_i \mapsto a_i$ given by the vector $a \in \mathbb{R}^{mn}$. We say that a variable matrix has a *rank-one sign pattern*, if for every assignment of the variables to $\{-1, +1\}$ the resulting sign matrix is rank one. A good example of a variable matrix with a rank-one sign pattern to keep in mind is $\left(\begin{smallmatrix} x_1 & x_2 \\ x_2 & x_1 \end{smallmatrix}\right)$. Notice that the matrices of the tensor product in Eq. (B.2.8) are of this form.

We claim the following:

Claim B.2.1. *Let A be m-by-n matrix, and B a variable matrix of the same size which has a rank-one sign pattern. Then for any assignment $a \in \mathbb{R}^{mn}$ to the variables of B*

$$\|A \circ B(a)\| \leq \|A\| \cdot \ell_\infty(a) \ . \tag{B.2.9}$$

Proof. Let u, v be two unit vectors and consider $|u^t(A \circ B(a))v|$. We wish to show that this is at most $\|A\| \cdot \ell_\infty(a)$. We will do this by forming a new assignment b, all of whose entries are in $\{-\ell_\infty(a), +\ell_\infty(a)\}$, and such that

$$|u^t(A \circ B(a))v| \leq |u^t(A \circ B(b))v| \ . \tag{B.2.10}$$

The result will then follow from the previous claim, since $B(b)$ is a constant times a rank-one sign matrix.

We form the assignment b as follows. Consider $u^t(A \circ B)v$ and break up this sum into two terms as $x_1 X + Y$ where Y does not contain factors of x_1. If $X(a) \geq 0$ then we set $b_1 = +\ell_\infty(a)$ and otherwise we set $b_1 = -\ell_\infty(a)$. In this way, $b_1 X(a) \geq a_1 X(a)$ and so $u^t(A \circ B(b_1, a_2, \ldots, a_{mn}))v \geq u^t(A \circ B(a_1, a_2, \ldots, a_{mn}))v$. We continue this process in turn isolating the variables x_2, x_3 and so on. ∎

Our claim now follows as the tensor product of matrices with rank-one sign patterns will again have a rank-one sign pattern. □

Vita

Rajat Mittal

2011 Ph. D. in Computer Science, Rutgers University

2000-2004 B. Tech. from Indian Institute of Technology, Bombay.

2000 Graduated from Instrumentation Limited School, India.

2010-2011 Teaching assistant, Department of Computer Science, Rutgers University.

2007-2010 Graduate assistant, Department of Computer Science, Rutgers University.

2004-2007 Teaching assistant, Department of Computer Science, Rutgers University.