

Parallel approximation of non-interactive zero-sum quantum games

Rahul Jain
Centre for Quantum Technologies
and Department of Computer Science
National University of Singapore

John Watrous
Institute for Quantum Computing
and School of Computer Science
University of Waterloo

Abstract

This paper studies a simple class of zero-sum games played by two competing quantum players: each player sends a mixed quantum state to a referee, who performs a joint measurement on the two states to determine the players' payoffs. We prove that an equilibrium point of any such game can be approximated by means of an efficient parallel algorithm, which implies that one-turn quantum refereed games, wherein the referee is specified by a quantum circuit, can be simulated in polynomial space.

1. Introduction

The theory of games has been studied extensively in mathematics and in several other disciplines for which it has applications. In theoretical computer science, computational aspects of game theory represent an important focus of the field.

There are several settings of interest to quantum computation and quantum cryptography that are naturally modeled by *quantum games*, which are games involving the exchange and processing of quantum information. For instance, multi-prover quantum interactive proofs [22], [8], [19], [20] can be modeled as cooperative quantum games; quantum coin-flipping [1], [21], [27], [24], [25] is naturally modeled as a game between two players that directly exchange quantum information; and quantum refereed games [13], [14], [15] are competitive games that model quantum interactive proofs with competing provers.

In this paper we consider a simple type of non-interactive, zero-sum quantum game: two competing players (hereafter called *Alice* and *Bob*) each send a mixed quantum state to a *referee*, who performs a joint measurement on the two states to determine the players' payoffs. For a fixed description of the referee, let $\phi(\rho, \sigma)$ denote Alice's expected payoff when she sends a mixed state ρ to the referee and Bob sends a mixed state σ . (For zero-sum games, Bob's expected payoff is then given by $-\phi(\rho, \sigma)$.) The theory of quantum information requires the function $\phi(\rho, \sigma)$ to be bilinear, from which it follows that

$$\max_{\rho} \min_{\sigma} \phi(\rho, \sigma) = \min_{\sigma} \max_{\rho} \phi(\rho, \sigma) \quad (1)$$

from well-known variants [9] of von Neumann's Min-Max Theorem [29]. (Indeed, such a fact holds for a much more general class of zero-sum quantum games that can allow for many rounds of interaction among the referee and players [15].) The value represented by the two sides of the equation (1) is called the *value* of the game. An *equilibrium point* of such a game is a pair of quantum states (ρ, σ) such that

$$\max_{\rho'} \phi(\rho', \sigma) = \phi(\rho, \sigma) = \min_{\sigma'} \phi(\rho, \sigma'),$$

the existence of which follows from the equation (1). In other words, when one player plays one of the states of an equilibrium point, the other has no incentive to play a state different from the other state in the pair. (These notions are, of course, similar to those for classical zero-sum games, but with some technical differences due to the nature of quantum information. In particular, there is a continuum of pure strategies for quantum players, corresponding to what are known as pure quantum states.) An equilibrium point of a zero-sum quantum game, given as an explicit description of the referee's measurement, can be efficiently computed by means of semidefinite programming.

The main result of this paper is an efficient *parallel* algorithm to find approximate equilibrium points of non-interactive zero-sum quantum games. For the case where the referee is specified by a quantum circuit rather than in an explicit matrix form, this algorithm implies that the value of such a game can be approximated in polynomial space. More succinctly, it implies that the complexity class QRG(1) of problems having one-turn quantum refereed games is contained in PSPACE.

Our algorithm uses the *multiplicative weights update method*, which has been described in several papers (such as [3] and [28], for instance), and which is explained in detail in the PhD thesis of S. Kale [18]. This general method captures many previously discovered algorithms, and has origins in learning theory, game theory, and optimization. At a technical level, our algorithm may accurately be described as being fairly straightforward within the context of this method, and in particular as it relates to online learning and game theory. It has, for instance, a close connection to the algorithm described in Section 3.2 of [18] for a particular type of matrix game; although our formulation of games

is somewhat different and corresponds more directly to a natural quantum information-theoretic notion of games. Both algorithms can be seen as non-commutative extensions of a well-known method of Freund and Schapire [11] to compute optimal strategies of classical zero-sum games. At a more specific level, our algorithm is based on unpublished work of Rohit Khandekar and the first author (Rahul Jain) for the performing this task.

What is most important about our specific algorithm, particularly for its application to the class QRG(1) of problems having one-turn refereed quantum games, is that it gives a sufficiently accurate solution and simultaneously can be *parallelized*. To our knowledge this issue of parallelizability has not been a concern in previous work on non-commutative applications of the multiplicative weights update method.

As mentioned above, the task of finding equilibrium strategies in zero-sum quantum games can be solved using semidefinite programming, even for interactive, multiple-round games [15], mimicking a similar relationship between classical zero-sum games and linear programming. Existing algorithms for solving semidefinite programs using the multiplicative weights update method, such as the ones described in [28], [2] and [4], might also potentially be parallelized and applied to the problem of finding equilibrium strategies in zero-sum quantum games—but given the simplicity of our algorithm, it would seem that the primary motivation for following this approach is to identify other tasks that may be performed in parallel.

Following this line of thought, one is naturally led to the question of what sorts of semidefinite programs can efficiently be approximated in parallel. Of course this is not possible for all semidefinite programs unless $\text{NC} = \text{P}$, but one reasonable class of semidefinite programs to consider is the class of *positive* semidefinite programs, which are defined in an analogous way to positive linear programs. (A precise definition appears in Section 5.) The idea that positive instances of semidefinite programs might be approximated in parallel is also suggested by the well-known parallel algorithm for approximating positive *linear* programs of Luby and Nisan [23]. This algorithm, which indeed falls into the category of multiplicative weights update algorithms, was further extended by Young [30].

We have not been able to extend the Luby–Nisan algorithm to positive semidefinite programs, primarily due to difficulties arising from the noncommutative nature of the problem. On the other hand, our algorithm for approximating equilibrium strategies does give a parallel algorithm for solving positive instances of semidefinite programs—but unfortunately the accuracy of this approach is very limited for some instances of positive semidefinite programs. This method is discussed in Section 5, which is intended mainly to illustrate the limitations of this approach and to suggest the general problem of approximating classes of semidefinite programs in parallel as an interesting direction

for future research.

Finally, we note that in work done in collaboration with Sarvagya Upadhyay, and subsequent to the initial submission of this paper, we have applied the multiplicative weights update method to a particular class of (positive) semidefinite programs to obtain the containment $\text{QIP}(2) \subseteq \text{PSPACE}$, where $\text{QIP}(2)$ denotes the class of problems having two-message quantum interactive proof systems [17]. The algorithm that achieves this containment turns out to be rather different from the one in this paper, and does not to our knowledge imply the results reported herein.

The remainder of this paper is organized as follows. First, in Section 2, we briefly discuss some necessary background information on quantum information theory, and define the sort of zero-sum quantum game that is studied in later sections. Then, in Section 3, we describe our algorithm and present its analysis, including an analysis of its precision requirements. Section 4 discusses the application of this algorithm to obtain the containment $\text{QRG}(1) \subseteq \text{PSPACE}$, and Section 5 explains its (limited) application to solving positive instances of semidefinite programs. The paper concludes with Section 6.

2. Preliminaries and definitions

This section gives a brief summary of the quantum information-theoretic concepts that are needed in the paper, and then defines non-interactive zero-sum quantum games. A few additional definitions that will be helpful later in the paper are also discussed.

2.1. Basic quantum information-theoretic notions

In this paper we require just a few basic concepts about quantum information; so it is not necessarily required that the reader has any prior familiarity with quantum information.

When we refer to a *quantum register* we simply mean a discrete quantum system that we wish to consider, such as a collection of qubits representing a message transmitted from one party to another. With any quantum register we associate some vector space $\mathcal{X} = \mathbb{C}^n$, for a positive integer n that intuitively represents the maximum number of distinct classical states that could be stored in the register without error. A *state* of such a register is represented by a *density matrix*, which is a $n \times n$ positive semidefinite matrix having trace equal to 1. Density matrices may reasonably be viewed as the quantum information-theoretic analogue to a vector of probabilities, representing a probability distribution. We will write $\text{D}(\mathcal{X})$ to denote the set of all density matrices associated with a register that is described by \mathcal{X} . It is natural to view such density matrices as *linear operators* acting on \mathcal{X} , and for this reason the term *density operator* is commonly used in place of *density matrix*.

When two registers having associated spaces $\mathcal{X} = \mathbb{C}^n$ and $\mathcal{Y} = \mathbb{C}^m$ are considered as a single compound register, the associated space becomes the tensor product space $\mathcal{X} \otimes \mathcal{Y} = \mathbb{C}^{nm}$. If the two registers are independently prepared in states described by ρ and σ , respectively, then the joint state is described by the $nm \times nm$ density matrix $\rho \otimes \sigma$. This matrix may be written in block form as

$$\rho \otimes \sigma = \begin{pmatrix} \rho_{1,1}\sigma & \cdots & \rho_{1,n}\sigma \\ \vdots & \ddots & \vdots \\ \rho_{n,1}\sigma & \cdots & \rho_{n,n}\sigma \end{pmatrix}.$$

In general, for a vector space $\mathcal{X} = \mathbb{C}^n$, we write $L(\mathcal{X})$ to denote the set of all $n \times n$ complex matrices, or linear operators mapping \mathcal{X} to itself, and we write $\text{Herm}(\mathcal{X})$ to refer to the subset of $L(\mathcal{X})$ given by the *Hermitian* matrices. These are the matrices A satisfying $A = A^*$, where A^* denotes the *adjoint* or *conjugate transpose* of A . The set $\text{Herm}(\mathcal{X})$ forms a vector space over \mathbb{R} , and many optimization methods designed for real-valued symmetric matrices extend to $\text{Herm}(\mathcal{X})$ with little or no special consideration. Finally, we write $\text{Pos}(\mathcal{X})$ to denote the subset of $\text{Herm}(\mathcal{X})$ that consists of all *positive semidefinite* $n \times n$ matrices (or operators acting on \mathcal{X}).

The *Hilbert-Schmidt inner product* on $L(\mathcal{X})$ is defined as

$$\langle A, B \rangle = \text{Tr}(A^*B)$$

for all $A, B \in L(\mathcal{X})$. It holds that $\langle A, B \rangle$ is a real number for all choices of Hermitian matrices A and B , and is a nonnegative real number for all choices of positive semidefinite matrices A and B .

A *measurement* of a register, having an associated vector space $\mathcal{X} = \mathbb{C}^n$, is described by a collection of $n \times n$ positive semidefinite matrices that sum to the identity. Specifically, a measurement that has some finite, non-empty set Σ of possible outcomes is described by a collection

$$\{P_a : a \in \Sigma\} \subset \text{Pos}(\mathcal{X})$$

satisfying

$$\sum_{a \in \Sigma} P_a = \mathbb{I}_{\mathcal{X}}.$$

Here, $\mathbb{I}_{\mathcal{X}}$ denotes the $n \times n$ identity matrix, or identity operator on \mathcal{X} . (The subscript \mathcal{X} is dropped when it is implicitly clear.) If the register corresponding to \mathcal{X} is in a state described by the density matrix $\rho \in D(\mathcal{X})$, and the measurement described by $\{P_a : a \in \Sigma\} \subset \text{Pos}(\mathcal{X})$ is performed, each outcome $a \in \Sigma$ will be observed with probability $\langle P_a, \rho \rangle$.

2.2. Non-interactive zero-sum quantum games

In a non-interactive zero-sum quantum game, Alice and Bob each send a quantum state to a referee, who performs a measurement on these two states to determine their payoffs.

Hereafter we will let $\mathcal{A} = \mathbb{C}^n$ and $\mathcal{B} = \mathbb{C}^m$ refer to the vector spaces corresponding to the states that Alice and Bob send to the referee.

When the referee performs a measurement to determine Alice and Bob's payoffs, a *joint* measurement is used. In other words, Alice's and Bob's states are together viewed as a single state of a register. We therefore have that the referee's measurement is described by a collection

$$\{R_a : a \in \Sigma\} \subset \text{Pos}(\mathcal{A} \otimes \mathcal{B})$$

that satisfies the condition

$$\sum_{a \in \Sigma} R_a = \mathbb{I}_{\mathcal{A} \otimes \mathcal{B}}.$$

If Alice sends the state ρ and Bob sends the state σ , then each possible measurement outcome $a \in \Sigma$ appears with probability $\langle R_a, \rho \otimes \sigma \rangle$.

A payoff for each player is associated with each possible measurement outcome $a \in \Sigma$. As we consider only zero-sum games, it is sufficient to describe these payoffs by a function $v : \Sigma \rightarrow \mathbb{R}$; with Alice's payoff for outcome a being $v(a)$ and Bob's payoff being $-v(a)$. For a given choice of states ρ and σ , it holds that Alice's expected payoff is given by

$$\sum_{a \in \Sigma} v(a) \langle R_a, \rho \otimes \sigma \rangle = \langle R, \rho \otimes \sigma \rangle$$

for

$$R = \sum_{a \in \Sigma} v(a) R_a.$$

Bob's expected payoff is given by $-\langle R, \rho \otimes \sigma \rangle$. When one is interested only in the expected payoff of a given game, it is therefore sufficient to consider that the game is simply determined by R . We will refer to R as a *payoff observable*, given that a matrix that arises in this way from a measurement and a real-valued function on its outcomes is sometimes called an observable.

A necessary and sufficient condition for a matrix R acting on $\mathcal{A} \otimes \mathcal{B}$ to arise from some measurement and real-valued payoff function v as just described is that R is Hermitian. The sort of payoff function $\phi(\rho, \sigma)$ discussed in the introduction therefore takes the form $\phi(\rho, \sigma) = \langle R, \rho \otimes \sigma \rangle$ for R ranging over the set of Hermitian matrices of the appropriate size. As the tensor product is a universal bilinear function, and every real-valued linear function on $\text{Herm}(\mathcal{A}) \otimes \text{Herm}(\mathcal{B})$ can be expressed as an inner product with some Hermitian matrix R , we have that a necessary and sufficient condition for $\phi(\rho, \sigma)$ to be a physically valid payoff function is that ϕ is a real-valued bilinear function.

Now, given that the sets $D(\mathcal{A})$ and $D(\mathcal{B})$ are convex and compact, and that Alice's expected payoff $\langle R, \rho \otimes \sigma \rangle$ is a bilinear function on $D(\mathcal{A}) \times D(\mathcal{B})$, it follows from well-known extensions [9] of von Neumann's Min-Max Theorem [29] that

$$\max_{\rho} \min_{\sigma} \langle R, \rho \otimes \sigma \rangle = \min_{\sigma} \max_{\rho} \langle R, \rho \otimes \sigma \rangle, \quad (2)$$

where the maximum and minimum are over $\rho \in \mathcal{D}(\mathcal{A})$ and $\sigma \in \mathcal{D}(\mathcal{B})$, respectively. We let $\alpha(R)$ denote the *value* of the game determined by R , which is the quantity represented by the two sides of the above equation (2). A pair of quantum states (ρ, σ) is called an *equilibrium point* for R if both ρ and σ independently achieve the maximum and minimum, respectively, in equation (2); or, equivalently, that

$$\min_{\sigma' \in \mathcal{D}(\mathcal{B})} \langle R, \rho \otimes \sigma' \rangle = \langle R, \rho \otimes \sigma \rangle = \max_{\rho' \in \mathcal{D}(\mathcal{A})} \langle R, \rho' \otimes \sigma \rangle.$$

Again, the existence of an equilibrium point follows easily from equation (2).

Finally, we define that an ϵ -*approximate equilibrium point* of a game with payoff observable R is a pair of states (ρ, σ) such that

$$\begin{aligned} \max_{\rho' \in \mathcal{D}(\mathcal{A})} \langle R, \rho' \otimes \sigma \rangle - \epsilon \|R\| \\ \leq \langle R, \rho \otimes \sigma \rangle \leq \min_{\sigma' \in \mathcal{D}(\mathcal{B})} \langle R, \rho \otimes \sigma' \rangle + \epsilon \|R\|. \end{aligned}$$

Note that this is an approximation in an additive sense, and is relative to the maximum absolute value of the expected payoff for the game (which is reflected by the presence of the factor $\|R\|$ in the error).

2.3. Additional definitions and notation

This section summarizes some additional terminology and notation that will be used in the paper. First, a linear mapping of the form $\Phi : \mathcal{L}(\mathcal{B}) \rightarrow \mathcal{L}(\mathcal{A})$ is called a *super-operator* (as it maps linear operators to linear operators). The *adjoint super-operator* to Φ has the form $\Phi^* : \mathcal{L}(\mathcal{A}) \rightarrow \mathcal{L}(\mathcal{B})$, and is uniquely determined by the condition

$$\langle A, \Phi(B) \rangle = \langle \Phi^*(A), B \rangle$$

for all $A \in \mathcal{L}(\mathcal{A})$ and $B \in \mathcal{L}(\mathcal{B})$. A super-operator $\Phi : \mathcal{L}(\mathcal{B}) \rightarrow \mathcal{L}(\mathcal{A})$ is said to be *positive* if it holds that $\Phi(P)$ is positive semidefinite for every choice of a positive semidefinite operator $P \in \text{Pos}(\mathcal{B})$. It is the case that Φ^* is positive if and only if Φ is positive.

There is a one-to-one and onto linear correspondence between the collection of operators of the form $R \in \mathcal{L}(\mathcal{A} \otimes \mathcal{B})$ and the collection of super-operators of the form $\Phi : \mathcal{L}(\mathcal{B}) \rightarrow \mathcal{L}(\mathcal{A})$, which is sometimes known as the *Choi-Jamiołkowski isomorphism*. Specifically, for every super-operator $\Phi : \mathcal{L}(\mathcal{B}) \rightarrow \mathcal{L}(\mathcal{A})$, one defines an operator $R \in \mathcal{L}(\mathcal{A} \otimes \mathcal{B})$, called the Choi-Jamiołkowski representation of Φ , by the equation

$$R = \sum_{1 \leq i, j \leq m} \Phi(E_{i,j}) \otimes E_{i,j},$$

where $E_{i,j}$ is the matrix with a 1 in entry (i, j) and 0 in every other entry. Conversely, given an operator $R \in \mathcal{L}(\mathcal{A} \otimes \mathcal{B})$,

one defines a super-operator $\Phi : \mathcal{L}(\mathcal{B}) \rightarrow \mathcal{L}(\mathcal{A})$ by means of the formula

$$\Phi(B) = \text{Tr}_{\mathcal{B}}(R(\mathbb{I}_{\mathcal{A}} \otimes B^{\top})). \quad (3)$$

It follows that

$$\text{Tr}(R(A \otimes B)) = \text{Tr}(A \Phi(B^{\top}))$$

for every choice of $A \in \mathcal{L}(\mathcal{A})$ and $B \in \mathcal{L}(\mathcal{B})$. These correspondences are both linear, and are inverse to one another—so one is free to translate back and forth between the two as necessary for a given application. If R is positive semidefinite, then the corresponding super-operator Φ is positive. (In fact, Φ has the stronger property of being *completely positive* if and only if R is positive semidefinite.)

For a given quantum game, we may equally well calculate expected payoffs and equilibrium points by using the unique super-operator Φ determined by (3) rather than the payoff observable R . In particular, (ρ, σ^{\top}) is an equilibrium point of the game defining R if and only if

$$\min_{\sigma' \in \mathcal{D}(\mathcal{B})} \langle \rho, \Phi(\sigma') \rangle = \langle \rho, \Phi(\sigma) \rangle = \max_{\rho' \in \mathcal{D}(\mathcal{A})} \langle \rho', \Phi(\sigma) \rangle, \quad (4)$$

and the value of this game is alternately expressed as

$$\begin{aligned} \alpha(\Phi) &= \max_{\rho \in \mathcal{D}(\mathcal{A})} \min_{\sigma \in \mathcal{D}(\mathcal{B})} \langle \rho, \Phi(\sigma) \rangle \\ &= \min_{\sigma \in \mathcal{D}(\mathcal{B})} \max_{\rho \in \mathcal{D}(\mathcal{A})} \langle \rho, \Phi(\sigma) \rangle. \end{aligned} \quad (5)$$

For a Hermitian $n \times n$ matrix A , one denotes the eigenvalues of A by

$$\lambda_1(A) \geq \lambda_2(A) \geq \dots \geq \lambda_n(A),$$

sorted from largest to smallest and including each eigenvalue a number of times equal to its multiplicity. For every $n \times n$ Hermitian matrix A , the *spectral norm* is denoted $\|A\|$ and satisfies

$$\|A\| = \max\{|\lambda_1(A)|, \dots, |\lambda_n(A)|\},$$

while the *trace norm* is denoted $\|A\|_1$ and satisfies

$$\|A\|_1 = |\lambda_1(A)| + \dots + |\lambda_n(A)|.$$

(Note that both of these formulas assume that A is Hermitian.)

Using the above notation, we may express the equations (4) and (5) in simpler terms: (ρ, σ^{\top}) is an equilibrium point of the game defining Φ if and only if

$$\lambda_1(\Phi(\sigma)) = \langle \rho, \Phi(\sigma) \rangle = \lambda_m(\Phi^*(\rho)),$$

while the value of this game satisfies

$$\alpha(\Phi) = \min_{\sigma \in \mathcal{D}(\mathcal{B})} \lambda_1(\Phi(\sigma)) = \max_{\rho \in \mathcal{D}(\mathcal{A})} \lambda_m(\Phi^*(\rho)).$$

Finally, for future reference we note that if a payoff observable R satisfies $0 \leq R \leq \mathbb{I}$, then it holds that $0 \leq \Phi(\sigma) \leq \mathbb{I}$ and $0 \leq \Phi^*(\rho) \leq \mathbb{I}$ for all choices of density matrices $\rho \in \mathcal{D}(\mathcal{A})$ and $\sigma \in \mathcal{D}(\mathcal{B})$. Moreover, for arbitrary Hermitian matrices $A \in \text{Herm}(\mathcal{A})$ and $B \in \text{Herm}(\mathcal{B})$, we have $\|\Phi(B)\| \leq \|B\|_1$ and $\|\Phi^*(A)\| \leq \|A\|_1$.

3. The Main Result

We now present the main result of the paper, which is a parallel algorithm to approximate the value of a non-interactive zero-sum quantum game. This fact is stated as Theorem 1 below, following a few comments on the assumed form of the input.

We suppose that a non-interactive zero-sum quantum game is given as a payoff observable $R \in \text{Herm}(\mathcal{A} \otimes \mathcal{B})$, for $\mathcal{A} = \mathbb{C}^n$ and $\mathcal{B} = \mathbb{C}^m$ as discussed in the previous section. More precisely, we assume that R is given as an $nm \times nm$ matrix, along with a specification of the dimensions n and m . Each entry of R is a complex number, which we assume has rational real and imaginary parts, each represented as the ratio of two integers expressed in binary notation. We let k be the maximum length of the binary representation over all of these integers, and define $\text{size}(R)$ to be $(nm)^2 k$. It is clear that $O(\text{size}(R))$ bits suffice to encode R .

In addition to R , n and m , an accuracy parameter $\varepsilon > 0$ is also given as input. For technical reasons it is most convenient to assume that ε is represented in *unary notation*: each string 1^r , for a positive integer r , denotes the value $\varepsilon = 1/r$. This assumption on the input form of ε reflects the fact that our algorithm does not scale well with respect to accuracy—it forces the length of the input to be proportional to $1/\varepsilon$ rather than $\log(1/\varepsilon)$, and therefore permits our algorithm to be described by circuits with size polynomial in the input length.

The output of the algorithm will be a pair of density matrices (ρ, σ) where $\rho \in \text{D}(\mathcal{A})$ and $\sigma \in \text{D}(\mathcal{B})$. They are assumed to be represented in a manner similar to the input matrix R .

Theorem 1. *An ε -approximate equilibrium point (ρ, σ) for a given payoff observable R can be computed by a logarithmic-space uniform family of Boolean circuits having depth polynomial in $\log(\text{size}(R))$ and $1/\varepsilon$.*

3.1. Parallel algorithm for positive games

Our algorithm is most naturally described for the case that the payoff observable R satisfies $0 \leq R \leq \mathbb{I}$. We therefore begin with this case, which will imply Theorem 1 by an appropriate translation and rescaling of R . The algorithm is described in Figure 1.

3.1.1. Accuracy of the algorithm. In this subsection, the accuracy of the algorithm described in Figure 1 is analyzed. We note that a similar type of analysis has appeared in previous works on the multiplicative weights update method and its predecessors [18].

At this point in the analysis we are concerned only with the idealized algorithm described in Figure 1—numerical

Algorithm

- 1) Let $\mu = \varepsilon/8$ and let $N = \lceil 64 \ln(nm)/\varepsilon^2 \rceil$.
- 2) Initialize: $A_0 = \mathbb{I}_{\mathcal{A}}$, $\rho_0 = A_0/\text{Tr}(A_0)$, $B_0 = \mathbb{I}_{\mathcal{B}}$, and $\sigma_0 = B_0/\text{Tr}(B_0)$.
- 3) For each j from 1 to N , let A_j , ρ_j , B_j , and σ_j be as follows:

$$\begin{aligned} A_j &= \exp\left(\mu \sum_{i=0}^{j-1} \Phi(\sigma_i)\right), \\ \rho_j &= A_j / \text{Tr}(A_j), \\ B_j &= \exp\left(-\mu \sum_{i=0}^{j-1} \Phi^*(\rho_i)\right), \\ \sigma_j &= B_j / \text{Tr}(B_j). \end{aligned}$$

- 4) Output the pair (ρ, σ^T) , where

$$\rho = \frac{1}{N} \sum_{j=0}^{N-1} \rho_j \quad \text{and} \quad \sigma = \frac{1}{N} \sum_{j=0}^{N-1} \sigma_j.$$

Figure 1. A parallel algorithm to compute an ε -approximate equilibrium point of a one-round zero-sum quantum game. The game is assumed to be described by a payoff observable R satisfying $0 \leq R \leq \mathbb{I}$, which gives rise to a positive map $\Phi : \text{L}(\mathcal{B}) \rightarrow \text{L}(\mathcal{A})$ as described in Section 2.3.

issues concerning the required precision with which the idealized operations are performed are discussed in the next subsection. We begin by noting some facts concerning matrix exponentials. First, the *Golden-Thompson Inequality* (see Section IX.3 of [5]) states that, for any two Hermitian matrices X and Y of equal dimension, we have

$$\text{Tr}(e^{X+Y}) \leq \text{Tr}(e^X e^Y).$$

Second is the following simple pair of inequalities concerning the matrix exponential of positive and negative semidefinite matrices.

Lemma 2. *Let P be an operator satisfying $0 \leq P \leq \mathbb{I}$. Then for every real number $\mu > 0$, the following two inequalities hold:*

$$\begin{aligned} \exp(\mu P) &\leq \mathbb{I} + \mu \exp(\mu) P, \\ \exp(-\mu P) &\leq \mathbb{I} - \mu \exp(-\mu) P. \end{aligned}$$

Proof. It is sufficient to prove the inequalities for P replaced by a scalar $\lambda \in [0, 1]$, for then the operator inequalities follow by considering a spectral decomposition of P . If $\lambda = 0$ both inequalities are immediate, so let us assume $\lambda > 0$. By the Mean Value Theorem there exists a value $\lambda_0 \in (0, \lambda)$

such that

$$\frac{\exp(\mu\lambda) - 1}{\lambda} = \mu \exp(\mu\lambda_0) \leq \mu \exp(\mu),$$

from which the first inequality follows. Similarly, there exists a value $\lambda_0 \in (0, \lambda)$ such that

$$\frac{\exp(-\mu\lambda) - 1}{\lambda} = -\mu \exp(-\mu\lambda_0) \leq -\mu \exp(-\mu),$$

which yields the second inequality. \square

We now proceed to the main part of the accuracy analysis, which comprises two bounds on the eigenvalues of $\Phi^*(\rho)$ and $\Phi(\sigma)$, where (ρ, σ^\top) is the output of the algorithm.

Lemma 3. *The following inequalities hold:*

$$\lambda_1(\Phi(\sigma)) \leq \frac{\exp(\mu)}{N} \sum_{j=1}^N \langle \rho_{j-1}, \Phi(\sigma_{j-1}) \rangle + \frac{\ln(n)}{\mu N},$$

$$\lambda_m(\Phi^*(\rho)) \geq \frac{\exp(-\mu)}{N} \sum_{j=1}^N \langle \rho_{j-1}, \Phi(\sigma_{j-1}) \rangle - \frac{\ln(m)}{\mu N}.$$

Proof. Let us begin by noting that each of the operators A_j and B_j (for $j = 0, \dots, N$) that are obtained during the course of the algorithm are positive definite, and therefore have positive trace. It follows that each of the operators ρ_j and σ_j is a well-defined density operator.

To prove the first inequality, observe that

$$A_N = \exp\left(\mu \sum_{j=1}^N \Phi(\sigma_{j-1})\right) = \exp(\mu N \Phi(\sigma)).$$

Given that A_N is positive definite, it holds that

$$\text{Tr}(A_N) \geq \lambda_1(A_N) = \exp(\mu N \lambda_1(\Phi(\sigma))),$$

and therefore

$$\lambda_1(\Phi(\sigma)) \leq \frac{\ln(\text{Tr}(A_N))}{\mu N}. \quad (6)$$

The first inequality in the statement of the lemma will now follow by bounding $\ln(\text{Tr}(X_N))$, which can be done as follows. First, note that we may alternately write

$$A_j = \exp(\ln(A_{j-1}) + \mu \Phi(\sigma_{j-1}))$$

for each $j \geq 1$, and therefore

$$\begin{aligned} \text{Tr}(A_j) &= \text{Tr}(\exp(\ln(A_{j-1}) + \mu \Phi(\sigma_{j-1}))) \\ &\leq \text{Tr}(A_{j-1} \exp(\mu \Phi(\sigma_{j-1}))) \end{aligned}$$

by the Golden-Thompson inequality. As σ_{j-1} is a density operator, it holds that $\Phi(\sigma_{j-1}) \leq \mathbb{I}$, and therefore

$$\exp(\mu \Phi(\sigma_{j-1})) \leq \mathbb{I} + \mu \exp(\mu) \Phi(\sigma_{j-1})$$

by Lemma 2. Thus, using the fact that $\text{Tr}(XY_1) \leq \text{Tr}(XY_2)$ for all choices of matrices X , Y_1 , and Y_2 with $X \geq 0$ and $Y_1 \leq Y_2$, we have

$$\begin{aligned} \text{Tr}(A_j) &\leq \text{Tr}(A_{j-1} (\mathbb{I} + \mu \exp(\mu) \Phi(\sigma_{j-1}))) \\ &= \text{Tr}(A_{j-1} (1 + \mu \exp(\mu) \langle \rho_{j-1}, \Phi(\sigma_{j-1}) \rangle)). \end{aligned}$$

It now follows from the inequality $1 + \mu \leq \exp(\mu)$ that

$$\text{Tr}(A_j) \leq \text{Tr}(A_{j-1}) \exp(\mu \exp(\mu) \langle \rho_{j-1}, \Phi(\sigma_{j-1}) \rangle).$$

Applying this inequality recursively, and using the fact that $\text{Tr}(A_0) = n$, we obtain

$$\text{Tr}(A_N) \leq n \cdot \exp\left(\mu \exp(\mu) \sum_{j=1}^N \langle \rho_{j-1}, \Phi(\sigma_{j-1}) \rangle\right). \quad (7)$$

Combining (6) and (7) yields

$$\lambda_1(\Phi(\sigma)) \leq \frac{\exp(\mu)}{N} \sum_{j=1}^N \langle \rho_{j-1}, \Phi(\sigma_{j-1}) \rangle + \frac{\ln(n)}{\mu N},$$

as required.

The second inequality follows by similar reasoning, except with a few differences that we now highlight. We first observe that

$$B_N = \exp(-\mu N \Phi^*(\rho)).$$

This time we have

$$\text{Tr}(B_N) \geq \lambda_1(B_N) = \exp(-\mu N \lambda_m(\Phi^*(\rho))),$$

where the switch from the largest eigenvalue to the smallest is caused by the minus sign in the exponential function. Thus,

$$\lambda_m(\Phi^*(\rho)) \geq -\frac{\ln(\text{Tr}(B_N))}{\mu N}. \quad (8)$$

The quantity $\text{Tr}(B_N)$ is now bounded in the same way as $\text{Tr}(A_N)$, except that we need the second inequality in Lemma 2. Specifically, the Golden-Thompson inequality implies

$$\begin{aligned} \text{Tr}(B_j) &= \text{Tr}(\exp(\ln(B_{j-1}) - \mu \Phi^*(\rho_{j-1}))) \\ &\leq \text{Tr}(B_{j-1} \exp(-\mu \Phi^*(\rho_{j-1}))). \end{aligned}$$

As $\Phi^*(\rho_{j-1}) \leq \mathbb{I}$ we have

$$\exp(-\mu \Phi^*(\rho_{j-1})) \leq \mathbb{I} - \mu \exp(-\mu) \Phi^*(\rho_{j-1}),$$

and therefore

$$\text{Tr}(B_j) \leq \text{Tr}(B_{j-1}) \exp(-\mu \exp(-\mu) \langle \sigma_{j-1}, \Phi^*(\rho_{j-1}) \rangle).$$

It follows that

$$\text{Tr}(B_N) \leq m \cdot \exp\left(-\mu \exp(-\mu) \sum_{j=1}^N \langle \sigma_{j-1}, \Phi^*(\rho_{j-1}) \rangle\right).$$

Combining this inequality with (8), and using the fact that $\langle \sigma_j, \Phi^*(\rho_j) \rangle = \langle \rho_j, \Phi(\sigma_j) \rangle$ for every choice of j , we have

$$\lambda_m(\Phi^*(\rho)) \geq \frac{\exp(-\mu)}{N} \sum_{j=1}^N \langle \rho_{j-1}, \Phi(\sigma_{j-1}) \rangle - \frac{\ln(m)}{\mu N},$$

which completes the proof. \square

It is now possible to verify that the output (ρ, σ^\top) of the algorithm satisfies

$$\begin{aligned} \max_{\rho' \in \mathcal{D}(A)} \langle R, \rho' \otimes \sigma^\top \rangle - \varepsilon/2 \\ \leq \langle R, \rho \otimes \sigma^\top \rangle \leq \min_{\sigma' \in \mathcal{D}(B)} \langle R, \rho \otimes \sigma' \rangle + \varepsilon/2, \end{aligned}$$

which is expressed in terms of the mapping Φ as

$$\lambda_1(\Phi(\sigma)) - \varepsilon/2 \leq \langle \rho, \Phi(\sigma) \rangle \leq \lambda_m(\Phi^*(\rho)) + \varepsilon/2.$$

It follows from Lemma 3 that

$$\begin{aligned} \lambda_1(\Phi(\sigma)) - \lambda_m(\Phi^*(\rho)) \\ \leq \frac{\exp(\mu) - \exp(-\mu)}{N} \sum_{j=1}^N \langle \sigma_{j-1}, \Phi(\rho_{j-1}) \rangle + \frac{\ln(nm)}{\mu N}, \end{aligned}$$

and given that each of the quantities $\langle \sigma_{j-1}, \Phi(\rho_{j-1}) \rangle$ is at most 1, we have

$$\begin{aligned} \lambda_1(\Phi(\sigma)) - \lambda_m(\Phi^*(\rho)) \\ \leq 2 \sinh(\mu) + \frac{\ln(nm)}{\mu N} < 3\mu + \frac{\ln(nm)}{\mu N} \leq \varepsilon/2. \end{aligned}$$

Thus, given that $\lambda_m(\Phi^*(\rho)) \leq \langle \rho, \Phi(\sigma) \rangle \leq \lambda_1(\Phi(\sigma))$, we have

$$\begin{aligned} \lambda_1(\Phi(\sigma)) - \varepsilon/2 \\ \leq \lambda_m(\Phi^*(\rho)) \leq \langle \rho, \Phi(\sigma) \rangle \leq \lambda_1(\Phi(\sigma)) \\ \leq \lambda_m(\Phi(\sigma)) + \varepsilon/2 \quad (9) \end{aligned}$$

as claimed.

3.1.2. Numerical precision and complexity of the algorithm.

Let us now consider the complexity of the algorithm described in Figure 1. It is the goal of this section to demonstrate that this algorithm can be implemented, by a logarithmic-space uniform family of Boolean circuits with depth polynomial in $\log(\text{size}(R)) + 1/\varepsilon$, with sufficient accuracy to obtain an ε -approximate equilibrium point for the input payoff observable R . Throughout the analysis, we (sometimes grossly) overestimate errors for the sake of simpler expressions involving as few variables as possible.

Each iteration performed in step 3 of the algorithm requires the evaluation of Φ and Φ^* , two matrix exponential computations, and a constant number of elementary matrix operations (in this case: addition, scalar multiplication, and

computation of the trace). Were it not for the matrix exponentials, it would be straightforward to perform all of the required operations within the claimed size and depth bounds using exact computations. Given that the matrix exponentials will generate irrational numbers, however, we must settle for approximations over the course of the algorithm. To guarantee that the algorithm is sufficiently accurate, it will suffice to perform all computations to within an additive error of $(\varepsilon/2) \exp(-8N^2)$, as is shown below. (We could afford to take a much smaller error with respect to $\text{size}(R)$, but there is no need to do this.)

Let us begin by making a few simple observations about the matrices computed throughout the course of the algorithm. The matrices $\sigma_0, \dots, \sigma_{N-1}$ are density matrices, and therefore it holds that $\|\Phi(\sigma_j)\| \leq 1$ for each choice of $j = 0, \dots, N-1$. Likewise, $\|\Phi^*(\rho_j)\| \leq 1$ for each choice of $j = 0, \dots, N-1$. Consequently, we have $\|A_j\| \leq e^N$, $\|B_j\| \leq e^N$, $1 \leq \text{Tr}(A_j) \leq e^{2N}$, and $e^{-N} \leq \text{Tr}(B_j) \leq e^N$ for $j = 0, \dots, N-1$.

Next, let us represent the actual matrices computed during the course of the algorithm by placing a tilde over the variables representing the idealized values that are expressed in Figure 1. It will suffice to prove that $\|\rho - \tilde{\rho}\|_1 \leq \varepsilon/2$ and $\|\sigma - \tilde{\sigma}\|_1 \leq \varepsilon/2$, for then the inequalities

$$|\lambda_1(\Phi(\sigma)) - \lambda_1(\Phi(\tilde{\sigma}))| \leq \frac{\varepsilon}{2}$$

and

$$|\lambda_m(\Phi^*(\rho)) - \lambda_m(\Phi^*(\tilde{\rho}))| \leq \frac{\varepsilon}{2}$$

hold. Combined with (9), we obtain the required accuracy.

Now, each iteration of step 3 of the algorithm will introduce some error into the calculation of the final answer. Let us consider the j -th iteration, and assume that a positive real number $\delta_j \in (0, 1)$ is given such that $\|\sigma_i - \tilde{\sigma}_i\|_1 \leq \delta_j$ for $i = 0, \dots, j-1$. Let us define

$$X_j = \mu \sum_{i=0}^{j-1} \Phi(\sigma_i) \quad \text{and} \quad \tilde{X}_j = \mu \sum_{i=0}^{j-1} \Phi(\tilde{\sigma}_i).$$

Then $\|X_j - \tilde{X}_j\| \leq \delta_j N$ and $\|X_j\| \leq N$, and therefore

$$\begin{aligned} \left\| \exp(X_j) - \exp(\tilde{X}_j) \right\| \\ \leq \|X_j - \tilde{X}_j\| \exp\left(\|X_j - \tilde{X}_j\|\right) \exp(\|X_j\|) < \delta_j e^{3N}, \end{aligned}$$

where the first of the two inequalities follows from a perturbation bound on the matrix exponential function. (In particular, we have used Corollary 6.2.32 of [16].) By computing the matrix exponential with accuracy δ_j we therefore have

$$\|A_j - \tilde{A}_j\| \leq \delta_j e^{4N},$$

and thus

$$\|A_j - \tilde{A}_j\|_1 \leq \delta_j e^{5N}.$$

It follows that

$$\begin{aligned} & \|\rho_j - \tilde{\rho}_j\|_1 \\ & \leq \frac{1}{\text{Tr}(A_j)} \|A_j - \tilde{A}_j\|_1 + \|\tilde{A}_j\|_1 \left| \frac{1}{\text{Tr}(A_j)} - \frac{1}{\text{Tr}(\tilde{A}_j)} \right| \\ & \leq \delta_j e^{8N}. \end{aligned}$$

By similar reasoning, if it holds that $\|\rho_i - \tilde{\rho}_i\|_1 \leq \delta_j$ for $i = 0, \dots, j-1$, then

$$\begin{aligned} & \|\sigma_j - \tilde{\sigma}_j\|_1 \\ & \leq \frac{1}{\text{Tr}(B_j)} \|B_j - \tilde{B}_j\|_1 + \|\tilde{B}_j\|_1 \left| \frac{1}{\text{Tr}(B_j)} - \frac{1}{\text{Tr}(\tilde{B}_j)} \right| \\ & \leq \delta_j e^{8N}. \end{aligned}$$

We conclude from these bounds that taking

$$\delta_j = (\varepsilon/2)e^{-8N^2}$$

guarantees that $\|\rho - \tilde{\rho}\|_1 \leq \varepsilon/2$ and $\|\sigma - \tilde{\sigma}\|_1 \leq \varepsilon/2$.

The required precision for the matrix exponentials is easily obtained by taking sufficiently many terms in the series

$$e^X = \mathbb{I} + X + \frac{1}{2}X^2 + \frac{1}{6}X^3 + \dots$$

(This of course is not the most efficient way to compute matrix exponentials, but it suffices to prove the main theorem.) For instance, taking $9N^2$ terms guarantees that the required accuracy $(\varepsilon/2)e^{-8N^2}$ is achieved.

At this point, the parallel complexity of the algorithm is easily bounded. Each of the matrices stored by the algorithm has entries whose real and imaginary parts are represented in binary notation using $O(N^2)$ bits. For each iteration in step 3 of the algorithm, the evaluations of Φ and Φ^* , as well as the elementary matrix operations, may therefore be performed by standard parallel algorithms (see, for instance, [12]) by logarithmic-space uniform Boolean circuits (with size that is necessarily polynomial in $\text{size}(R)$ and $1/\varepsilon$ given this uniformity constraint), within depth that is polynomial in $\log(\text{size}(R))$ and $1/\varepsilon$. The number of iterations performed is N , which results in total depth polynomial in $\log(\text{size}(R))$ and $1/\varepsilon$.

3.2. Extensions to arbitrary payoff observables

For an arbitrary payoff observable R , the algorithm from the previous section is not guaranteed to function correctly, as we have used the positivity of the corresponding superoperator Φ several times during the analysis.

It is straightforward, however, to translate and scale an arbitrary payoff observable in a way that allows the algorithm to be used. For an arbitrary positive semidefinite payoff observable R , this is essentially trivial—one simply

runs the algorithm on the payoff observable $P = R/\|R\|$. For a negative semidefinite payoff observable R , one simply exchanges the roles of Alice and Bob and considers the payoff observable $-R$ (with the spaces \mathcal{A} and \mathcal{B} swapped).

Let us now consider the general case of a payoff observable R for which $\lambda_1(R) > 0 > \lambda_{nm}(R)$. Define

$$P = \frac{R - \lambda_{nm}(R)\mathbb{I}}{\lambda_1(R) - \lambda_{nm}(R)}.$$

Then $0 \leq P \leq \mathbb{I}$, and so the algorithm from the previous section may be used to obtain an ε -approximation (ρ, σ) for P . The point (ρ, σ) is easily verified to be a δ -approximate equilibrium point for R , where

$$\delta = \frac{\lambda_1(R) - \lambda_{nm}(R)}{\|R\|} \varepsilon \leq 2\varepsilon.$$

4. QRG(1) is contained in PSPACE

Quantum interactive proof systems with two competing provers are naturally represented as games between two competing players, moderated by a referee. The two players (Alice and Bob) play the roles of competing provers, while the referee corresponds to the verifier. Quantum refereed games have been studied in [14], [13], [15], and represent a quantum analogue to the classical refereed games model studied in [10].

The simplest form of a refereed quantum game has the general form defined in Section 2; meaning that there is no communication from the referee to the players. The players each send a quantum state to the referee, who performs a joint measurement to determine the winner. With this picture in mind, one defines the complexity class QRG(1) to be the class consisting of all promise problems $A = (A_{\text{yes}}, A_{\text{no}})$ for which there exists a polynomial-time uniform family $Q = \{Q_n : n \in \mathbb{N}\}$ of quantum circuits, where each circuit Q_n takes $n + 2p(n)$ input qubits for some polynomial bounded function p , such that the following properties hold:

- For every string $x \in A_{\text{yes}}$ it holds that

$$\max_{\rho} \min_{\sigma} \Pr[Q(x, \rho, \sigma) = 1] \geq \frac{2}{3}.$$

- For every string $x \in A_{\text{no}}$ it holds that

$$\max_{\rho} \min_{\sigma} \Pr[Q(x, \rho, \sigma) = 1] \leq \frac{1}{3}.$$

Here, the maximum and minimum are both over all quantum states on $p(|x|)$ qubits, and the notation $Q(x, \rho, \sigma) = 1$ is shorthand for the event that a measurement of some fixed output qubit of the circuit $Q_{|x|}$ (with respect to the standard basis) yields 1, assuming that the input to the circuit is the state $|x\rangle\langle x| \otimes \rho \otimes \sigma$. The name QRG(1) refers to the fact that these are quantum refereed games with 1 turn, during which the players send quantum states to the referee in parallel.

The class QRG(1) may be viewed as a simple variant of QMA, where there are two competing provers rather than a single prover. It is obvious that $\text{QMA} \subseteq \text{QRG}(1)$, and that QRG(1) is closed under complementation (and thus $\text{co-QMA} \subseteq \text{QRG}(1)$).

The class QRG(1) may roughly be thought of as a quantum analogue to the class S_2^P that was defined by [7] and [26], and it is easily observed that $S_2^P \subseteq \text{QRG}(1)$. There is one subtlety, however, which is that the definition of QRG(1) does not allow one prover to see the other prover's message (which would not make sense in the quantum setting anyway), whereas the standard definition of S_2^P does.

Proposition 4. $\text{QRG}(1) \subseteq \text{PSPACE}$.

Proof. Suppose that A is a promise problem contained in QRG(1), and that $\{Q_n\}$ is a polynomial-time uniform family of quantum circuits that witnesses this fact. For each input x , let R_x denote the payoff observable that corresponds to the game played by the players Alice and Bob on input x , where the payoff for Alice is defined as 1 for acceptance and 0 for rejection. The expected payoff is therefore the probability of acceptance, which Alice tries to maximize and Bob tries to minimize.

We denote by $\text{NC}(\text{poly})$ the class of promise problems computed by polynomial-space uniform Boolean circuits with polynomial depth. It holds that $\text{NC}(\text{poly}) \subseteq \text{PSPACE}$ [6], so it therefore suffices to prove that $A \in \text{NC}(\text{poly})$. This is easily accomplished by composing three families of Boolean circuits:

- 1) A family of Boolean circuits that outputs a description of the payoff observable R_x associated with the game on input x .
- 2) The family of Boolean circuits given by Theorem 1, that finds an ε -approximate equilibrium point (ρ, σ) of the payoff observable R_x , for $\varepsilon = 1/8$.
- 3) A family of Boolean circuits that computes the expected payoff $\langle R_x, \rho \otimes \sigma \rangle$, and accepts if the value is greater than $1/2$ (rejecting otherwise).

The first family is easily derived from the circuits $\{Q_n\}$, by computing the product of a polynomial number of exponential-size matrices that correspond to the quantum gates of the appropriate circuit Q_n . This family may be taken to be polynomial-space uniform, with polynomial depth. The second family is, as suggested above, given by Theorem 1. This family is logarithmic-space uniform and has polynomial-size and poly-logarithmic depth with respect to $\text{size}(R_x)$. Thus, with respect to the input length $|x|$, this family is polynomial-space uniform, and has polynomial depth. The last family is required only to perform elementary matrix and arithmetic operations, and can be taken to have similar properties as the first two: polynomial-space uniformity and polynomial depth. Composing these families in the appropriate way demonstrates that $A \in \text{NC}(\text{poly})$ as required. \square

5. Parallel Approximation of Positive SDPs

We now discuss the connection between equilibrium points of non-interactive zero-sum quantum games and semidefinite programs. The main focus of this section will, in particular, be on *positive* instances of semidefinite programs, and on the question of whether good parallel methods to approximate them exist. We will first discuss the general notion of positive instances of semidefinite programs and then explain how our algorithm may be used in their approximation, albeit with poor accuracy in some cases.

The multiplicative weights update method has been applied to semidefinite programming in [2], [4], and the general connection between equilibrium points of different types of games and linear/semidefinite programs is well-known. Once again, the reader is referred to Kale [18] for further details and historical remarks.

5.1. Positive SDPs in super-operator form

Suppose that the following input has been given:

- 1) a Hermitian matrix $A \in \mathbb{C}^{n \times n}$,
- 2) a Hermitian matrix $B \in \mathbb{C}^{m \times m}$, and
- 3) a linear mapping $\Phi : \mathbb{C}^{m \times m} \rightarrow \mathbb{C}^{n \times n}$ (i.e., a super-operator) that preserves Hermiticity.

To say that Φ preserves Hermiticity means that $\Phi(Y)$ is Hermitian for every choice of a Hermitian matrix $Y \in \mathbb{C}^{m \times m}$. This condition is equivalent to the Choi-Jamiołkowski representation R of Φ being a Hermitian matrix.

Given this input, let us consider the following semidefinite programming problem, which we say is in the *super-operator form*:

Primal problem	Dual problem
maximize: $\langle B, Y \rangle$	minimize: $\langle A, X \rangle$
subject to: $\Phi(Y) \leq A,$	subject to: $\Phi^*(X) \geq B,$
$Y \geq 0.$	$X \geq 0.$

Here, X and Y range over all (positive semidefinite) matrices in $\mathbb{C}^{n \times n}$ and $\mathbb{C}^{m \times m}$, respectively. This form is completely general: it is possible to translate semidefinite programs in so-called *standard form* to the super-operator form, and vice versa. It can be shown that strong duality holds for semidefinite programs in the super-operator form under conditions that are similar to those for standard form semidefinite programs. In particular, the existence of either of the following implies that strong duality holds:

- 1) a positive definite matrix Y for which $\Phi(Y) < A$, or
- 2) a positive definite matrix X for which $\Phi^*(X) > B$.

We define that such a problem instance is *positive* if A and B are positive semidefinite matrices and Φ is a positive super-operator. Let us also define that such a problem is *strictly positive* if it holds that A and B are positive definite

and Φ is a strictly positive super-operator (which means that $\Phi(\mathbb{I})$ is positive definite). Strong duality necessarily holds for all strictly positive semidefinite programs in the super-operator form.

5.2. Parallel approximation of strictly positive SDPs

As discussed in the introduction, the algorithm from Section 3.1 can be used to approximate some positive instances of semidefinite programs in parallel. We consider only strictly positive semidefinite programs, and we note that an arbitrary strictly positive semidefinite program in the super-operator form can be transformed into one of the following simpler form:

<u>Primal problem</u>	<u>Dual problem</u>
maximize: $\text{Tr}(Y)$	minimize: $\text{Tr}(X)$
subject to: $\Phi(Y) \leq \mathbb{I},$ $Y \geq 0.$	subject to: $\Phi^*(X) \geq \mathbb{I},$ $X \geq 0.$

This may be done by defining

$$\Phi(Y) = A^{-\frac{1}{2}} \Psi \left(B^{-\frac{1}{2}} Y B^{-\frac{1}{2}} \right) A^{-\frac{1}{2}}$$

for a given problem instance defined by $A > 0$, $B > 0$, and a strictly positive super-operator Ψ .

Now, to make the connection with the algorithm from Section 3 clear, let us recall that we define $\mathcal{A} = \mathbb{C}^n$ and $\mathcal{B} = \mathbb{C}^m$, and suppose that the super-operator Φ that represents the above semidefinite program takes the form $\Phi : \mathbb{L}(\mathcal{B}) \rightarrow \mathbb{L}(\mathcal{A})$. Let us also define $\text{opt}(\Phi)$ to be the optimal value of the primal problem (which is the same as the optimal value of the dual problem by strong duality). It is clear that $\text{opt}(\Phi) > 0$, for some positive scalar multiple of the identity must be primal feasible. Let us also recall that we have defined

$$\alpha(\Phi) = \max_{\rho \in \mathbb{D}(\mathcal{A})} \min_{\sigma \in \mathbb{D}(\mathcal{B})} \langle \rho, \Phi(\sigma) \rangle.$$

Proposition 5. *For all strictly positive super-operators Φ we have $\alpha(\Phi) = 1/\text{opt}(\Phi)$.*

Proof. Let (ρ, σ) be an equilibrium point of Φ , meaning that

$$\lambda_1(\Phi(\sigma)) = \langle \rho, \Phi(\sigma) \rangle = \lambda_m(\Phi^*(\rho)) = \alpha(\Phi).$$

The assumption that Φ is strictly positive implies that $\alpha(\Phi)$ is positive.

We now observe that $\sigma/\alpha(\Phi)$ is primal feasible, as it is positive semidefinite and satisfies

$$\lambda_1(\Phi(\sigma/\alpha(\Phi))) = 1,$$

which implies $\Phi(\sigma/\alpha(\Phi)) \leq \mathbb{I}$. Likewise, $\rho/\alpha(\Phi)$ is dual feasible as it is positive semidefinite and satisfies

$$\lambda_m(\Phi^*(\rho/\alpha(\Phi))) = 1,$$

which implies $\Phi^*(\rho/\alpha(\Phi)) \geq \mathbb{I}$. Both result in the same objective value $1/\alpha(\Phi)$, and so the proposition follows by (weak) duality. \square

It follows that the algorithm from Section 3.1 may be used to approximate $\text{opt}(\Phi)$, albeit with limited accuracy for some choices of Φ , by taking the reciprocal of the value of the game associated with Φ . To be more specific, let R be the Choi-Jamiołkowski representation of the super-operator Φ as discussed in Section 2.3. Let us write $\tilde{\alpha}(\Phi)$ to denote the approximate value of the game described by R that results from the algorithm's ε -approximate equilibrium point of R , and let us also write

$$\widetilde{\text{opt}}(\Phi) = 1/\tilde{\alpha}(\Phi).$$

We then have

$$\left(1 - \frac{\varepsilon \|R\|}{\alpha(\Phi)}\right) \alpha(\Phi) \leq \tilde{\alpha}(\Phi) \leq \left(1 + \frac{\varepsilon \|R\|}{\alpha(\Phi)}\right) \alpha(\Phi)$$

and therefore

$$\begin{aligned} \left(1 + \frac{\varepsilon \|R\|}{\alpha(\Phi)}\right)^{-1} \text{opt}(\Phi) \\ \leq \widetilde{\text{opt}}(\Phi) \leq \left(1 - \frac{\varepsilon \|R\|}{\alpha(\Phi)}\right)^{-1} \text{opt}(\Phi). \end{aligned}$$

For choices of Φ for which $\alpha(\Phi)$ is large and $\|R\|$ is small (bounded below and above by constants, say), a reasonable approximation to $\text{opt}(\Phi)$ may be obtained. For many choices of Φ , however, our method is clearly not suitable, and we believe it is an interesting problem for future research to find more accurate parallel algorithms for this problem.

6. Conclusion

In this paper we have shown that equilibrium points of non-interactive zero-sum quantum games can be efficiently computed in parallel, using the multiplicative weights update method. As a consequence, we have that one-turn quantum refereed games can be simulated in polynomial space, or $\text{QRG}(1) \subseteq \text{PSPACE}$. We have also illustrated the connection between values of quantum games and positive instances of semidefinite programming problems.

The main open question that we wish to raise concerns the existence of efficient parallel algorithms for positive instances of semidefinite programming problems. The class of such problems for which our algorithm gives accurate solutions is limited. To what extent can this task be performed more accurately and more generally?

Acknowledgements

The parallel algorithm presented in this work was inspired by an algorithm appearing in an unpublished joint work, concerning classical games, of Rohit Khandekar and the first

author. We therefore thank Rohit Khandekar for his implicit and indirect contribution to this work, and for allowing us to include it in this paper. Rahul Jain’s research is supported by ARO/NSA USA, and John Watrous’s research is supported by Canada’s NSERC and the Canadian Institute for Advanced Research.

References

- [1] A. Ambainis. A new protocol and lower bounds for quantum coin flipping. In *Proceedings of the Thirty-Third Annual ACM Symposium on Theory of Computing*, pages 134–142, 2001.
- [2] S. Arora, E. Hazan, and S. Kale. Fast algorithms for approximate semidefinite programming using the multiplicative weights update method. In *Proceedings of the 46th Annual IEEE Symposium on Foundations of Computer Science*, pages 339–348, 2005.
- [3] S. Arora, E. Hazan, and S. Kale. The multiplicative weights update method: a meta algorithm and applications. Manuscript, 2005.
- [4] S. Arora and S. Kale. A combinatorial, primal-dual approach to semidefinite programs. In *Proceedings of the Thirty-Ninth Annual ACM Symposium on Theory of Computing*, pages 227–236, 2007.
- [5] R. Bhatia. *Matrix Analysis*. Springer, 1997.
- [6] A. Borodin. On relating time and space to size and depth. *SIAM Journal on Computing*, 6:733–744, 1977.
- [7] R. Canetti. On BPP and the polynomial-time hierarchy. *Information Processing Letters*, 57:237–241, 1996.
- [8] R. Cleve, P. Høyer, B. Toner, and J. Watrous. Consequences and limits of nonlocal strategies. In *Proceedings of the 19th Annual IEEE Conference on Computational Complexity*, pages 236–249, 2004.
- [9] K. Fan. Minimax theorems. *Proceedings of the National Academy of Sciences*, 39:42–47, 1953.
- [10] U. Feige and J. Kilian. Making games short. In *Proceedings of the Twenty-Ninth Annual ACM Symposium on Theory of Computing*, pages 506–516, 1997.
- [11] Y. Freund and R. Schapire. Game theory, on-line prediction and boosting. In *Proceedings of the Ninth Annual Conference on Computational Learning Theory*, pages 325–332, 1996.
- [12] J. von zur Gathen. Parallel linear algebra. In J. Reif, editor, *Synthesis of Parallel Algorithms*, chapter 13. Morgan Kaufmann Publishers, Inc., 1993.
- [13] G. Gutoski. Upper bounds for quantum interactive proofs with competing provers. In *Proceedings of the 20th Annual IEEE Conference on Computational Complexity*, pages 334–343, 2005.
- [14] G. Gutoski and J. Watrous. Quantum interactive proofs with competing provers. In *Proceedings of the 22nd Symposium on Theoretical Aspects of Computer Science*, volume 3404 of *Lecture Notes in Computer Science*, pages 605–616. Springer, 2005.
- [15] G. Gutoski and J. Watrous. Toward a general theory of quantum games. In *Proceedings of the 39th ACM Symposium on Theory of Computing*, pages 565–574, 2007.
- [16] R. Horn and C. Johnson. *Topics in Matrix Analysis*. Cambridge University Press, 1991.
- [17] R. Jain, S. Upadhyay, and J. Watrous. Two-message quantum interactive proofs are in PSPACE. Manuscript, 2009.
- [18] S. Kale. *Efficient Algorithms Using the Multiplicative Weights Update Method*. PhD thesis, Princeton University, 2007.
- [19] J. Kempe, H. Kobayashi, K. Matsumoto, B. Toner, and T. Vidick. Entangled games are hard to approximate. In *Proceedings of the 49th Annual IEEE Symposium on Foundations of Computer Science*, 2008.
- [20] J. Kempe, H. Kobayashi, K. Matsumoto, and T. Vidick. Using entanglement in quantum multi-prover interactive proofs. In *Proceedings of the 23rd Annual Conference on Computational Complexity*, 2008.
- [21] A. Kitaev. Quantum coin-flipping. Presentation at the 6th Workshop on *Quantum Information Processing (QIP 2003)*, 2002.
- [22] H. Kobayashi and K. Matsumoto. Quantum multi-prover interactive proof systems with limited prior entanglement. *Journal of Computer and System Sciences*, 66(3), 2003.
- [23] M. Luby and N. Nisan. A parallel approximation algorithm for positive linear programming. In *Proceedings of the 25th Annual ACM Symposium on Theory of Computing*, pages 448–457, 1993.
- [24] C. Mochon. Quantum weak coin-flipping with bias of 0.192. In *45th Annual IEEE Symposium on Foundations of Computer Science*, pages 2–11, 2004.
- [25] C. Mochon. Quantum weak coin flipping with arbitrarily small bias. Available as *arXiv.org e-print 0711.4114*, 2007.
- [26] A. Russell and R. Sundaram. Symmetric alternation captures BPP. *Computational Complexity*, 7:152–162, 1998.
- [27] R. Spekkens and T. Rudolph. Quantum protocol for cheat-sensitive weak coin flipping. *Physical Review Letters*, 89(22):227901, 2002.
- [28] K. Tsuda, G. Rätsch, and M. Warmuth. Matrix exponentiated gradient updates for on-line learning and Bregman projection. *The Journal of Machine Learning Research*, 6:995–1018, 2005.
- [29] J. von Neumann. Zur theorie der gesellschaftsspiele. *Mathematische Annalen*, 100(1928):295–320, 1928.
- [30] N. Young. Sequential and parallel algorithms for mixed packing and covering. In *Proceedings of the 42nd IEEE Symposium on Foundations of Computer Science*, pages 538–546, 2001.