QIP = PSPACE

Rahul Jain* Zhengfeng Ji† Sarvagya Upadhyay‡ John Watrous‡

*Department of Computer Science and Centre for Quantum Technologies
National University of Singapore
Republic of Singapore

†Perimeter Institute for Theoretical Physics
Waterloo, Ontario, Canada
and
State Key Laboratory of Computer Science
Institute of Software, Chinese Academy of Sciences
Beijing, China

‡Institute for Quantum Computing and School of Computer Science
University of Waterloo
Waterloo, Ontario, Canada

ABSTRACT

We prove that the complexity class QIP, which consists of all problems having quantum interactive proof systems, is contained in PSPACE. This containment is proved by applying a parallelized form of the matrix multiplicative weights update method to a class of semidefinite programs that captures the computational power of quantum interactive proofs. As the containment of PSPACE in QIP follows immediately from the well-known equality IP = PSPACE, the equality QIP = PSPACE follows.

Categories and Subject Descriptors

F.1.3 [**Theory of Computation**]: Computation by Abstract Devices—*Complexity Measures and Classes*

General Terms

Theory

Keywords

Quantum interactive proof systems, quantum computation, matrix multiplicative weights update method, semidefinite programming

1. INTRODUCTION

Efficient proof verification is a fundamental notion in computational complexity theory. The most direct complexity-theoretic abstraction of efficient proof verification is represented by the complexity class NP, wherein a deterministic polynomial-time verification procedure decides whether a polynomial-length proof string is valid for a given input. One cannot overstate the importance of this notion—it is essential to complexity theory and finds applications throughout the broad range of scientific fields that make use of the theory of NP-completeness.

In the mid-1980's, Babai [4] and Goldwasser, Micali, and Rackoff [21] introduced a computational model that extends the notion of efficient proof verification to an *interactive* setting. (Journal versions of these papers appeared later as [5] and [22].) In this model, which is known as the *interactive proof system* model, a computationally bounded *verifier* interacts with a *prover* of unlimited computation power. The interaction comprises one or more rounds of communication between the prover and verifier, and the verifier may make use of randomly generated bits during the interaction. After the rounds of communication are finished, the verifier makes a decision to *accept* or *reject* based on the interaction.

A decision problem A, which we may take to be a *promise problem* [15, 19] in the interest of generality, is said to have an interactive proof system if there exists a polynomial-time verifier that meets two conditions: the *completeness* condition and the *soundness* condition. The completeness condition formalizes the requirement that true statements can be proved, which in the present setting means that if an input string x is a yes-instance of A, then there exists a course of action for the prover that causes the verifier to accept with high probability. The soundness condition formalizes the requirement that false statements cannot be proved, meaning in this case that if an input string x is a no-instance of A, then the verifier will reject with high probability no matter what course of action the prover takes. One denotes by IP the collection of decision problems having interactive proof systems that meet these conditions.

The computational power of interactive proof systems was not known when they were first introduced, but it was soon determined to coincide with PSPACE, the class of problems solvable deterministically in polynomial space. The containment IP $\subseteq$ PSPACE is fairly straightforward—it was first proved in [17], and a proof may also be found in [43]. Known proofs of the reverse containment PSPACE $\subseteq$ IP, on the other hand, are not straightforward, and make essential use of the technique commonly known as *arithmetization* [33, 41, 42].

Many variants of interactive proof systems have been studied, including public-coin interactive proofs [4, 5, 23], zero-knowledge interactive proofs [21, 22, 20], multi-prover interactive proofs [7], and interactive proofs with competing provers [16].

The present paper is concerned with *quantum interactive proof systems*, which were first studied roughly a decade after IP = PSPACE was proved [45, 30]. The fundamental notions of this model are the same as those of classical interactive proof systems, except that the prover and verifier may now process and exchange quantum information. Similar to the classical case, several variants of quantum interactive proof systems have been studied, including those considered in [24, 25, 29, 32, 31, 35, 46].

The complexity class QIP is defined as the class of decision problems having complete and sound quantum interactive proof systems. QIP trivially contains IP, as the ability of a verifier to process quantum information is never a hindrance: a quantum verifier can simulate a classical verifier, and a single computationally unbounded prover can never use quantum information to an advantage against a verifier behaving classically. The inclusion PSPACE $\subseteq$ QIP is therefore immediate. The best upper bound on QIP known prior to the present paper was QIP $\subseteq$ EXP, where EXP denotes the class of problems solvable in deterministic exponential-time. This containment was proved in [30] through the use of semidefinite programming: the optimal probability with which a given verifier can be made to accept in a quantum interactive proof system can be represented as an exponential-size semidefinite program, and known polynomial-time algorithms for semidefinite programming provide the required tool to prove the containment. It has been an open problem for the last decade to establish more precise bounds on the class QIP.

It is known that quantum information offers at least one significant advantage to the interactive proof system model under a reasonable complexity-theoretic assumption. Specifically, quantum interactive proof systems can be *parallelized* to three messages, meaning that quantum interactive proof systems in which just three messages are exchanged between the prover and verifier already have the full power of quantum interactive proofs having a polynomial number of messages [30]. Classical interactive proofs cannot have this property unless the polynomial-time hierarchy collapses to the second level [5, 23].

Three of us [26] recently showed that QIP(2), the class of problem having 2-message quantum interactive proof systems, is contained in PSPACE. Our proof made use of a parallel algorithm, based on a method known as the *matrix multiplicative weights update method*, to approximate optimal solutions for a class of semidefinite programs that represent the maximum acceptance probabilities for verifiers in two-message quantum interactive proofs. In the present paper we extend this result to all of QIP, establishing the relationship QIP = PSPACE. Similar to [26], we use the matrix multiplicative weights update method, together with parallel methods for matrix computations.

The multiplicative weights method is a framework for algorithm design having its origins in various fields, including learning theory, game theory, and optimization. Its matrix variant, as discussed in the survey paper [2] and the PhD thesis of Kale [28], gives an iterative way to approximate the optimal value of semidefinite programs [3, 44]. In addition to its application in [26], it was applied to quantum complexity in [27] to prove the containment of a different quantum complexity class QRG(1) in PSPACE. The key strength of this method for these applications is that it can be parallelized for some special classes of semidefinite programs.

A key result that allows the matrix multiplicative weights update method to be applied to the entire class QIP is a characterization of QIP in terms of *quantum Arthur–Merlin games* that was proved in [35]. This characterization is described in greater detail later in the paper.

2. PRELIMINARIES

We assume the reader has familiarity with complexity theory and the basics of quantum computing, and refer readers who do not to [1] and [39]. The purpose of this section is to clarify some of the notation, terminology, and background knowledge on quantum information and basic linear algebra that we use or assume later in the paper.

A *quantum register* refers simply to a collection of qubits that we wish to assign a name. The 2^n-dimensional complex vector space associated with an n-qubit quantum register X is denoted $\mathcal{X}$, and a similar convention is used for other register names. The space associated with a tuple of quantum registers is given by the tensor product of the spaces associated with the individual registers.

For any vector space $\mathcal{X}$ associated with one or more quantum registers and any two linear operators (or matrices) A and B mapping $\mathcal{X}$ to itself, we define the inner product

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

where A^* denotes the adjoint (or conjugate transpose) of A. (We will only refer to such inner products for Hermitian operators A and B, so one may safely view that $\langle A, B\rangle = \operatorname{Tr}(AB)$ throughout this paper.) We write $\operatorname{Pos}(\mathcal{X})$ to refer to the set of all positive semidefinite operators on $\mathcal{X}$, and $\operatorname{D}(\mathcal{X})$ to refer to the set of all density operators on $\mathcal{X}$. For Hermitian operators A and B, the notations $A \leq B$ and $B \geq A$ mean that $B - A$ is positive semidefinite. An operator $\Pi \in \operatorname{Pos}(\mathcal{X})$ is a *projection* if all of its eigenvalues are either 0 or 1. The identity operator on $\mathcal{X}$ is denoted $\mathbb{1}_{\mathcal{X}}$, or simply by $\mathbb{1}$ when $\mathcal{X}$ can safely be taken as implicit.

A *quantum state* of a quantum register X is a density operator $\rho \in \operatorname{D}(\mathcal{X})$, and a *measurement* on X is a collection

$$\{P_b : b \in \Gamma\} \subseteq \operatorname{Pos}(\mathcal{X})$$

satisfying

$$\sum_{b\in\Gamma} P_b = \mathbb{1}_{\mathcal{X}}.$$

The set Γ is the set of *measurement outcomes*, and when such a measurement is performed on X while it is in the state ρ, each outcome $b \in \Gamma$ occurs with probability $\langle P_b, \rho\rangle$. A *projective measurement* is one in which each P_b is a projection.

For spaces $\mathcal{X}$ and $\mathcal{Y}$, one defines the *partial trace* $\operatorname{Tr}_{\mathcal{Y}}$ to be the unique linear mapping that satisfies $\operatorname{Tr}_{\mathcal{Y}}(A \otimes B) = (\operatorname{Tr} B)A$ for all operators A on $\mathcal{X}$ and B on $\mathcal{Y}$. A similar notation is used for the partial trace over $\mathcal{X}$ rather than $\mathcal{Y}$, as well as for partial traces defined on three or more tensor factors. When a pair of registers (X, Y) is viewed as a single register and has the quantum state $\rho \in \operatorname{D}(\mathcal{X} \otimes \mathcal{Y})$, one defines the state of X to be $\operatorname{Tr}_{\mathcal{Y}}(\rho)$. In other words, the partial trace describes the action of destroying, or simply ignoring, a given quantum register.

The *spectral norm* of an operator A mapping $\mathcal{X}$ to itself is defined as

$$\|A\| = \max\{\|Ax\| \,:\, x \in \mathcal{X},\, \|x\| = 1\},$$

where $\|Ax\|$ and $\|x\|$ refer to the Euclidean norm on $\mathcal{X}$. The spectral norm is sub-multiplicative: $\|AB\| \leq \|A\|\,\|B\|$ for all operators A and B. It holds that $\|P\|$ is equal to the largest eigenvalue of P for every positive semidefinite operator P.

For any operator A, the exponential of A is defined as

$$\exp(A) = \mathbb{1} + A + A^2/2 + A^3/6 + \cdots$$

The *Golden-Thompson Inequality* (see Section IX.3 of [8]) states that, for any two Hermitian operators A and B on $\mathcal{X}$, we have

$$\operatorname{Tr}\left[\exp(A+B)\right] \leq \operatorname{Tr}\left[\exp(A)\exp(B)\right].$$

3. SEMIDEFINITE PROGRAMS FOR QIP

The first step in our proof that QIP $\subseteq$ PSPACE is the identification of a family of semidefinite programs based on quantum interactive proof systems with the following form:

1. On any given input string x, the prover begins by sending a quantum register X to the verifier.
2. Upon receiving X from the prover, the verifier generates a bit $a \in \{0,1\}$ uniformly at random and sends this bit to the prover.
3. The prover responds with a second quantum register Y. (There is no loss of generality in assuming Y has the same number of qubits as X.)
4. The verifier performs one of two binary-valued measurements, determined by the value of the random bit a, on the pair (X,Y). A measurement outcome of 1 is interpreted as *acceptance*, while the outcome 0 is interpreted as *rejection*.

Such a proof system is an example of a *quantum Arthur-Merlin game* given that the verifier's messages to the prover consist entirely of uniformly generated random bits (which is analogous to the classical definition [4, 5]). It was proved in [35] that every problem $A \in$ QIP has a quantum interactive proof system of the above form, where the completeness is perfect and the soundness error is bounded by $1/2 - \varepsilon$ for any desired constant $\varepsilon > 0$. It may, in addition, be assumed that the measurements performed by the verifier in step 4 are projective measurements.

Now, for a given proof system of the above form, and for a fixed input string x, let us denote by $\mathcal{X}$ and $\mathcal{Y}$ the complex vector spaces corresponding to the registers X and Y, respectively. Let us suppose further that

$$\left\{\Pi_b^a \,:\, a, b \in \{0,1\}\right\} \subset \operatorname{Pos}\left(\mathcal{X} \otimes \mathcal{Y}\right)$$

are the projection operators on $\mathcal{X} \otimes \mathcal{Y}$ that describe the measurements performed by the verifier in step 4 (meaning that $\{\Pi_0^a, \Pi_1^a\}$ describes the binary-valued measurement that is performed for each $a \in \{0,1\}$). We claim that the maximum probability with which the prover can cause the verifier to accept (or output 1) in this proof system is given by the optimal value of the following semidefinite optimization problem:

$$\begin{aligned}
\text{maximize:}\quad & \frac{1}{2}\left\langle \Pi_1^0, \rho_0 \right\rangle + \frac{1}{2}\left\langle \Pi_1^1, \rho_1 \right\rangle \\
\text{subject to:}\quad & \operatorname{Tr}_{\mathcal{Y}}(\rho_0) = \operatorname{Tr}_{\mathcal{Y}}(\rho_1), \\
& \rho_0,\, \rho_1 \in \mathrm{D}\left(\mathcal{X} \otimes \mathcal{Y}\right).
\end{aligned}$$

The density operators ρ_0 and ρ_1 represent the possible states of the register pair (X,Y) conditioned on the verifier choosing $a = 0$ and $a = 1$, respectively. As one ranges over all actions for the prover, the possible state-pairs (ρ_0, ρ_1) that may result are precisely those states that agree on X, i.e., those satisfying

$$\operatorname{Tr}_{\mathcal{Y}}(\rho_0) = \operatorname{Tr}_{\mathcal{Y}}(\rho_1).$$

It is clear that this condition is necessary, as the prover sends X before learning a, while its sufficiency follows from the well-known principle in quantum information theory sometimes called the *unitary equivalence of purifications*.

Now, for the sake of the algorithm to be discussed in the next section, it is necessary that we consider a relaxation of the above semidefinite program. First, for any choice of $\alpha > 0$, let us introduce a block matrix

$$P_\alpha = \begin{pmatrix} \Pi_1^0 + \alpha \Pi_0^0 & 0 \\ 0 & \Pi_1^1 + \alpha \Pi_0^1 \end{pmatrix}, \tag{1}$$

whose inverse is given by

$$P_\alpha^{-1} = \begin{pmatrix} \Pi_1^0 + \frac{1}{\alpha}\Pi_0^0 & 0 \\ 0 & \Pi_1^1 + \frac{1}{\alpha}\Pi_0^1 \end{pmatrix}.$$

By taking $\mathcal{A}$ to be a complex vector space of dimension 2, which intuitively corresponds to the verifier's random bit a, we may view that $P_\alpha \in \operatorname{Pos}\left(\mathcal{A} \otimes \mathcal{X} \otimes \mathcal{Y}\right)$. Along similar lines, the density operators ρ_0 and ρ_1 may be combined into a single block density matrix

$$\rho = \begin{pmatrix} \frac{1}{2}\rho_0 & 0 \\ 0 & \frac{1}{2}\rho_1 \end{pmatrix} \in \mathrm{D}\left(\mathcal{A} \otimes \mathcal{X} \otimes \mathcal{Y}\right).$$

Finally, we introduce an auxiliary variable $\sigma \in \mathrm{D}\left(\mathcal{X}\right)$ in order to express the equality $\operatorname{Tr}_{\mathcal{Y}}(\rho_0) = \operatorname{Tr}_{\mathcal{Y}}(\rho_1)$ in the form of two inequalities. With these operators in mind, we consider the following semidefinite program:

$$\begin{aligned}
\text{maximize:}\quad & \left\langle P_\alpha^{-2}, \rho \right\rangle \\
\text{subject to:}\quad & \operatorname{Tr}_{\mathcal{Y}}(\rho) \leq \frac{1}{2}\mathbb{1}_{\mathcal{A}} \otimes \sigma, \\
& \rho \in \operatorname{Pos}\left(\mathcal{A} \otimes \mathcal{X} \otimes \mathcal{Y}\right), \\
& \sigma \in \mathrm{D}\left(\mathcal{X}\right).
\end{aligned}$$

This semidefinite program has an optimal value that is at least as large as the optimal value of the original problem, and is at most $1/\alpha^2$ plus that value.

An equivalent formulation of this problem, which we take to be the formal statement of the semidefinite program that is to be considered in the next section, is as follows:

<u>Primal problem</u>

$$\begin{aligned}
\text{maximize:}\quad & \operatorname{Tr}(\rho) \\
\text{subject to:}\quad & \operatorname{Tr}_{\mathcal{Y}}(P_\alpha \rho P_\alpha) \leq \frac{1}{2}\mathbb{1}_{\mathcal{A}} \otimes \sigma, \\
& \operatorname{Tr}(\sigma) \leq 1, \\
& \rho \in \operatorname{Pos}\left(\mathcal{A} \otimes \mathcal{X} \otimes \mathcal{Y}\right), \\
& \sigma \in \operatorname{Pos}\left(\mathcal{X}\right).
\end{aligned}$$

The dual problem takes the following form:

$$\underline{\text{Dual problem}}$$

$$\begin{aligned} \text{minimize:} \quad & \frac{1}{2}\left\|\operatorname{Tr}_{\mathcal{A}}(Q)\right\| \\ \text{subject to:} \quad & P_\alpha(Q \otimes \mathbb{1}_{\mathcal{Y}})P_\alpha \geq \mathbb{1}_{\mathcal{A}\otimes\mathcal{X}\otimes\mathcal{Y}}, \\ & Q \in \operatorname{Pos}(\mathcal{A}\otimes\mathcal{X}). \end{aligned}$$

4. A PARALLEL SDP ALGORITHM

We now present a parallel algorithm that operates as follows. It takes as input four projection operators

$$\Pi_0^0,\ \Pi_1^0,\ \Pi_0^1,\ \Pi_1^1 \in \operatorname{Pos}(\mathcal{X}\otimes\mathcal{Y})$$

satisfying $\Pi_0^0 + \Pi_1^0 = \mathbb{1}_{\mathcal{X}\otimes\mathcal{Y}} = \Pi_0^1 + \Pi_1^1$, where $\mathcal{X}$ and $\mathcal{Y}$ are complex vector spaces, both having dimension N. Under the promise that the semidefinite program described in the previous section has an optimal value that is either close to 1 or close to 1/2, it determines which is the case (accepting if the optimal value is close to 1 and rejecting when it is close to 1/2). The algorithm is as follows.

1. Let

   $$\alpha = 4, \quad \gamma = \frac{4}{3}, \quad \varepsilon = \frac{1}{64} \quad \text{and} \quad \delta = \frac{\varepsilon}{\alpha^2},$$

   and let P_α be as defined in (1). Set the number of iterations to be performed by the algorithm to

   $$T = \left\lceil \frac{8\log(N)}{\varepsilon^2\delta} \right\rceil$$

   and initialize

   $$\begin{aligned} X_0 &= \mathbb{1}_{\mathcal{A}\otimes\mathcal{X}\otimes\mathcal{Y}}, & \rho_0 &= X_0/\operatorname{Tr}(X_0), \\ Y_0 &= \mathbb{1}_{\mathcal{X}}, & \sigma_0 &= Y_0/\operatorname{Tr}(Y_0). \end{aligned}$$

2. Repeat the following steps for each $t = 0, \ldots, T-1$.

   (a) Let Δ_t be the projection operator onto the positive eigenspace of

   $$\operatorname{Tr}_{\mathcal{Y}}\left(P_\alpha \rho_t P_\alpha\right) - \frac{\gamma}{2}\mathbb{1}_{\mathcal{A}} \otimes \sigma_t,$$

   and let

   $$\beta_t = \langle P_\alpha(\Delta_t \otimes \mathbb{1}_{\mathcal{Y}})P_\alpha, \rho_t\rangle.$$

   If $\beta_t \leq \varepsilon$ then *halt and accept*.

   (b) Let

   $$X_{t+1} = \exp\left(-\varepsilon\delta \sum_{j=0}^{t} P_\alpha\left(\Delta_j \otimes \mathbb{1}_{\mathcal{Y}}\right)P_\alpha/\beta_j\right),$$

   $$Y_{t+1} = \exp\left(\varepsilon\delta \sum_{j=0}^{t} \operatorname{Tr}_{\mathcal{A}}\left(\Delta_j/\beta_j\right)\right),$$

   and let

   $$\rho_{t+1} = \frac{X_{t+1}}{\operatorname{Tr}(X_{t+1})} \quad \text{and} \quad \sigma_{t+1} = \frac{Y_{t+1}}{\operatorname{Tr}(Y_{t+1})}.$$

3. If acceptance did not occur in step 2, then *halt and reject*.

4.1 Analysis (ignoring precision)

We now show that the algorithm answers correctly, meaning that (i) if the optimal value of the semidefinite program from the previous section is sufficiently close to 1, then the algorithm accepts, and (ii) if the optimal value is sufficiently close to 1/2, then the algorithm rejects. We first do this in the idealized situation where all of the matrix operations are performed exactly, as this analysis better illustrates the basic principles of the algorithm. The finite-precision case is discussed in the subsection following this one.

Assume first that the algorithm accepts. This must happen during some iteration of step 2, and for whichever iteration t it is we will write ρ, σ, Δ, and β rather than ρ_t, σ_t, Δ_t, and β_t to simplify our notation. Define $\rho' \in \operatorname{Pos}(\mathcal{A}\otimes\mathcal{X}\otimes\mathcal{Y})$ and $\sigma' \in \operatorname{Pos}(\mathcal{X})$ as follows:

$$\begin{aligned} \rho' &= \frac{\rho}{\gamma + 4\beta}, \\ \sigma' &= \frac{\gamma\sigma + 4\operatorname{Tr}_{\mathcal{A}}\left[\Delta \operatorname{Tr}_{\mathcal{Y}}\left(P_\alpha\rho P_\alpha\right)\Delta\right]}{\gamma + 4\beta}. \end{aligned}$$

We will prove that (ρ', σ') is a primal feasible point of our semidefinite program that achieves an objective value significantly larger than 1/2. By the definition of Δ it holds that

$$\operatorname{Tr}_{\mathcal{Y}}(P_\alpha\rho P_\alpha) - \frac{\gamma}{2}\mathbb{1}_{\mathcal{A}}\otimes\sigma \leq \Delta\left(\operatorname{Tr}_{\mathcal{Y}}(P_\alpha\rho P_\alpha) - \frac{\gamma}{2}\mathbb{1}_{\mathcal{A}}\otimes\sigma\right)\Delta,$$

and given that σ is positive semidefinite it is immediate that

$$\Delta\left(\operatorname{Tr}_{\mathcal{Y}}(P_\alpha\rho P_\alpha) - \frac{\gamma}{2}\mathbb{1}_{\mathcal{A}}\otimes\sigma\right)\Delta \leq \Delta\operatorname{Tr}_{\mathcal{Y}}\left(P_\alpha\rho P_\alpha\right)\Delta.$$

Finally, by Lemma 1 (which is stated below) it holds that

$$\Delta\operatorname{Tr}_{\mathcal{Y}}(P_\alpha\rho P_\alpha)\Delta \leq 2\mathbb{1}_{\mathcal{A}}\otimes\operatorname{Tr}_{\mathcal{A}}\left[\Delta\operatorname{Tr}_{\mathcal{Y}}\left(P_\alpha\rho P_\alpha\right)\Delta\right],$$

and so we conclude that

$$\operatorname{Tr}_{\mathcal{Y}}(P_\alpha\rho P_\alpha) \leq \frac{1}{2}\mathbb{1}_{\mathcal{A}}\otimes\left(\gamma\sigma + 4\operatorname{Tr}_{\mathcal{A}}\left[\Delta\operatorname{Tr}_{\mathcal{Y}}\left(P_\alpha\rho P_\alpha\right)\Delta\right]\right).$$

It therefore holds that

$$\operatorname{Tr}_{\mathcal{Y}}(P_\alpha\rho' P_\alpha) \leq \frac{1}{2}\mathbb{1}_{\mathcal{A}}\otimes\sigma',$$

and it is clear that $\operatorname{Tr}(\sigma') = 1$, so (ρ', σ') is primal feasible as claimed. The objective value achieved by the point (ρ', σ') is given by

$$\operatorname{Tr}(\rho') \geq \frac{1}{\gamma + 4\beta} \geq \frac{1}{\gamma + 4\varepsilon} > \frac{5}{8}.$$

Now assume that the algorithm rejects, and define

$$Q = \frac{1 + 4\varepsilon}{T}\sum_{t=0}^{T-1}\Delta_t/\beta_t.$$

We will prove that Q is dual feasible and achieves a dual objective value that is significantly smaller than 1.

To prove the dual feasibility of Q, let us first observe that for each choice of $t = 0, \ldots, T-1$ we have

$$\operatorname{Tr}\left(X_{t+1}\right) \leq \operatorname{Tr}\left(X_t \exp\left(-\frac{\varepsilon\delta}{\beta_t}P_\alpha(\Delta_t\otimes\mathbb{1}_{\mathcal{Y}})P_\alpha\right)\right)$$

by the Golden-Thompson inequality. As we have assumed that the algorithm rejects, it holds that $\beta_t > \varepsilon$, and therefore

$$\left\|\frac{\delta}{\beta_t}P_\alpha(\Delta_t\otimes\mathbb{1}_{\mathcal{Y}})P_\alpha\right\| \leq \frac{\delta\alpha^2}{\varepsilon} \leq 1.$$

By Lemma 2 (which is also stated below) this implies that

$$\exp\left(-\frac{\varepsilon\delta}{\beta_t} P_\alpha(\Delta_t \otimes \mathbb{1}_{\mathcal{Y}})P_\alpha\right) \leq \mathbb{1} - \frac{\varepsilon\delta\exp(-\varepsilon)}{\beta_t} P_\alpha(\Delta_t \otimes \mathbb{1}_{\mathcal{Y}})P_\alpha,$$

and therefore

$$\begin{aligned}\operatorname{Tr}(X_{t+1}) &\leq \operatorname{Tr}\left(X_t\left(\mathbb{1} - \frac{\varepsilon\delta\exp(-\varepsilon)}{\beta_t}P_\alpha(\Delta_t \otimes \mathbb{1}_{\mathcal{Y}})P_\alpha\right)\right)\\ &\leq \operatorname{Tr}(X_t)\left(1 - \frac{\varepsilon\delta\exp(-\varepsilon)}{\beta_t}\langle P_\alpha(\Delta_t \otimes \mathbb{1}_{\mathcal{Y}})P_\alpha, \rho_t\rangle\right)\\ &= \operatorname{Tr}(X_t)(1-\varepsilon\delta\exp(-\varepsilon))\\ &\leq \operatorname{Tr}(X_t)\exp\left(-\varepsilon\delta\exp(-\varepsilon)\right),\end{aligned}$$

where the last inequality follows from $1+z \leq \exp(z)$, which holds for all real numbers z. By applying the above inequality recursively we obtain the inequality

$$\begin{aligned}\operatorname{Tr}(X_T) &\leq \operatorname{Tr}(X_0)\exp\left(-T\varepsilon\delta\exp(-\varepsilon)\right)\\ &= 2N^2\exp\left(-T\varepsilon\delta\exp(-\varepsilon)\right).\end{aligned}$$

On the other hand, we have that

$$\begin{aligned}\operatorname{Tr}(X_T) &= \operatorname{Tr}\left(\exp\left(-\varepsilon\delta\sum_{t=0}^{T-1}P_\alpha(\Delta_t\otimes\mathbb{1}_{\mathcal{Y}})P_\alpha/\beta_t\right)\right)\\ &\geq \exp\left(-\varepsilon\delta\lambda_{\min}\left(\sum_{t=0}^{T-1}P_\alpha(\Delta_t\otimes\mathbb{1}_{\mathcal{Y}})P_\alpha/\beta_t\right)\right),\end{aligned}$$

where the function $\lambda_{\min}$ is defined as the smallest eigenvalue of its argument. Thus, we have

$$\begin{aligned}&2N^2\exp(-T\varepsilon\delta\exp(-\varepsilon))\\ &\qquad\geq \exp\left(-\varepsilon\delta\lambda_{\min}\left(\sum_{t=0}^{T-1}P_\alpha\left(\Delta_t\otimes\mathbb{1}_{\mathcal{Y}}\right)P_\alpha/\beta_t\right)\right),\end{aligned}$$

and therefore

$$\begin{aligned}&\lambda_{\min}\left(\sum_{t=0}^{T-1}P_\alpha\left(\Delta_t\otimes\mathbb{1}_{\mathcal{Y}}\right)P_\alpha/\beta_t\right)\\ &\qquad\geq T\exp(-\varepsilon) - \frac{\log(2N^2)}{\varepsilon\delta}.\end{aligned}$$

Consequently

$$\begin{aligned}&\lambda_{\min}(P_\alpha(Q\otimes\mathbb{1}_{\mathcal{Y}})P_\alpha)\\ &\qquad\geq (1+4\varepsilon)\left(\exp(-\varepsilon) - \frac{\log(2N^2)}{T\varepsilon\delta}\right) \geq 1,\end{aligned}$$

and therefore $P_\alpha(Q\otimes\mathbb{1}_{\mathcal{Y}})P_\alpha \geq \mathbb{1}_{\mathcal{A}\otimes\mathcal{X}\otimes\mathcal{Y}}$, which implies that Q is dual feasible as claimed.

It remains to prove an upper bound on the dual objective value achieved by Q. For each $t = 0,\ldots,T-1$ we have

$$\operatorname{Tr}(Y_{t+1}) \leq \operatorname{Tr}\left(Y_t\exp\left(\varepsilon\delta\operatorname{Tr}_{\mathcal{A}}(\Delta_t/\beta_t)\right)\right).$$

We again make use of the inequality $\beta_t > \varepsilon$, this time to conclude that $\|\delta\operatorname{Tr}_{\mathcal{A}}\left(\Delta_t/\beta_t\right)\| < 1$. Using Lemma 2 again, we have

$$\exp\left(\varepsilon\delta\operatorname{Tr}_{\mathcal{A}}(\Delta_t/\beta_t)\right) \leq \mathbb{1} + \varepsilon\delta\exp(\varepsilon)\operatorname{Tr}_{\mathcal{A}}(\Delta_t/\beta_t).$$

Thus we have

$$\begin{aligned}\operatorname{Tr}(Y_{t+1}) &\leq \operatorname{Tr}(Y_t)\left(1+\varepsilon\delta\exp(\varepsilon)\langle\operatorname{Tr}_{\mathcal{A}}(\Delta_t/\beta_t),\sigma_t\rangle\right)\\ &= \operatorname{Tr}(Y_t)\left(1+\frac{\varepsilon\delta\exp(\varepsilon)}{\beta_t}\langle\Delta_t,\mathbb{1}_{\mathcal{A}}\otimes\sigma_t\rangle\right).\end{aligned}$$

We now make use of the fact that

$$\left\langle\Delta_t, \operatorname{Tr}_{\mathcal{Y}}(P_\alpha\rho_tP_\alpha) - \frac{\gamma}{2}\mathbb{1}_{\mathcal{A}}\otimes\sigma_t\right\rangle \geq 0,$$

which implies that

$$\langle\Delta_t,\mathbb{1}_{\mathcal{A}}\otimes\sigma_t\rangle \leq \frac{2}{\gamma}\langle\Delta_t,\operatorname{Tr}_{\mathcal{Y}}(P_\alpha\rho_tP_\alpha)\rangle = \frac{2\beta_t}{\gamma}.$$

We therefore have

$$\begin{aligned}\operatorname{Tr}(Y_{t+1}) &\leq \operatorname{Tr}(Y_t)\left(1+\frac{2\varepsilon\delta\exp(\varepsilon)}{\gamma}\right)\\ &\leq \operatorname{Tr}(Y_t)\exp\left(\frac{2\varepsilon\delta\exp(\varepsilon)}{\gamma}\right).\end{aligned}$$

By applying this inequality recursively we find that

$$\operatorname{Tr}(Y_T) \leq N\exp\left(\frac{2T\varepsilon\delta\exp(\varepsilon)}{\gamma}\right).$$

On the other hand we have

$$\begin{aligned}\operatorname{Tr}(Y_T) &= \operatorname{Tr}\left[\exp\left(\varepsilon\delta\sum_{t=0}^{T-1}\operatorname{Tr}_{\mathcal{A}}(\Delta_t/\beta_t)\right)\right]\\ &\geq \exp\left(\varepsilon\delta\left\|\sum_{t=0}^{T-1}\operatorname{Tr}_{\mathcal{A}}(\Delta_t/\beta_t)\right\|\right),\end{aligned}$$

and therefore

$$\left\|\sum_{t=0}^{T-1}\operatorname{Tr}_{\mathcal{A}}(\Delta_t/\beta_t)\right\| \leq \frac{2T\exp(\varepsilon)}{\gamma} + \frac{\log(N)}{\varepsilon\delta}.$$

The dual objective value obtained by Q therefore satisfies

$$\begin{aligned}\frac{1}{2}\|\operatorname{Tr}_{\mathcal{A}}(Q)\| &\leq (1+4\varepsilon)\left(\frac{\exp(\varepsilon)}{\gamma}+\frac{\log(N)}{2T\varepsilon\delta}\right)\\ &\leq \frac{1+8\varepsilon}{\gamma} < \frac{7}{8}.\end{aligned}$$

Now, to complete the analysis, suppose that the optimal value of the semidefinite program described in the previous section is greater than 7/8. Then it cannot be that the algorithm rejects, so the algorithm must accept. Similarly, if the optimal value is smaller than 5/8, then the algorithm cannot accept, so it must reject. These bounds are sufficient for the application of this algorithm to the containment of QIP in PSPACE to be presented in Section 5.

It remains to state and prove the two lemmas used in the analysis above.

LEMMA 1. *Let $R \in \operatorname{Pos}(\mathcal{A}\otimes\mathcal{Z})$ be a positive semidefinite operator, and assume* $\dim(\mathcal{A}) = 2$. *Then* $R \leq 2\mathbb{1}_{\mathcal{A}}\otimes\operatorname{Tr}_{\mathcal{A}}(R)$.

PROOF. Let σ_x, σ_y and σ_z denote the Pauli operators on $\mathcal{A}$. In matrix form they are

$$\sigma_x = \begin{pmatrix}0 & 1\\ 1 & 0\end{pmatrix},\quad \sigma_y = \begin{pmatrix}0 & -i\\ i & 0\end{pmatrix}\quad\text{and}\quad \sigma_z = \begin{pmatrix}1 & 0\\ 0 & -1\end{pmatrix}.$$

As each of these operators is Hermitian, we have that

$$\begin{gathered}(\sigma_x\otimes\mathbb{1}_{\mathcal{Z}})R(\sigma_x\otimes\mathbb{1}_{\mathcal{Z}}),\\ (\sigma_y\otimes\mathbb{1}_{\mathcal{Z}})R(\sigma_y\otimes\mathbb{1}_{\mathcal{Z}}),\\ (\sigma_z\otimes\mathbb{1}_{\mathcal{Z}})R(\sigma_z\otimes\mathbb{1}_{\mathcal{Z}})\end{gathered}$$

are all positive semidefinite. It therefore holds that

$$\begin{aligned} 2\mathbb{1}_{\mathcal{A}} \otimes \operatorname{Tr}_{\mathcal{X}}(R) &= R + (\sigma_x \otimes \mathbb{1}_{\mathcal{Z}}) R (\sigma_x \otimes \mathbb{1}_{\mathcal{Z}}) \\ &\quad + (\sigma_y \otimes \mathbb{1}_{\mathcal{Z}}) R (\sigma_y \otimes \mathbb{1}_{\mathcal{Z}}) \\ &\quad + (\sigma_z \otimes \mathbb{1}_{\mathcal{Z}}) R (\sigma_z \otimes \mathbb{1}_{\mathcal{Z}}) \\ &\geq R \end{aligned}$$

as required. □

LEMMA 2. *Let R be an operator satisfying $0 \leq R \leq \mathbb{1}$. Then for every real number $\eta > 0$, the following two inequalities hold:*

$$\begin{aligned} \exp(\eta R) &\leq \mathbb{1} + \eta \exp(\eta) R, \\ \exp(-\eta R) &\leq \mathbb{1} - \eta \exp(-\eta) R. \end{aligned}$$

PROOF. It is sufficient to prove the inequalities for R replaced by a scalar $\lambda \in [0, 1]$, for then the operator inequalities follow by considering a spectral decomposition of R. 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

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

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

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

which yields the second inequality. □

4.2 Comments on precision

The analysis in the previous subsection has assumed that all computations performed by the algorithm are exact. It is, however, necessary for our implementation of the algorithm to take some of the computations to be approximate. In particular, the computation of the positive eigenspace projections in step 2(a) and the matrix exponentials in step 2(b) will need to be approximate computations. Fortunately this is not a significant obstacle; when the approximate computations are performed with sufficient accuracy, a similar analysis to the one presented in the previous subsection shows that the correct answer is obtained. Here we provide a brief sketch of this analysis.

Based on the specific implementation of the algorithm to be discussed in the next subsection, we may make the following assumptions about the algorithm's accuracy.

1. For each $t = 0, \ldots, T-1$, Δ_t is a positive semidefinite operator and satisfies
$$\left\| \sqrt{\Delta_t} - \Lambda_t \right\| < \frac{\varepsilon^2}{8\alpha^2 N} \qquad (2)$$
for Λ_t being the true projection operator onto the positive eigenspace of $\operatorname{Tr}_{\mathcal{Y}}(P_\alpha \rho_t P_\alpha) - \frac{\gamma}{2}\mathbb{1}_{\mathcal{A}} \otimes \sigma_t$. Although it is not essential, we may also assume for the sake of convenience that each Δ_t satisfies $\Delta_t \leq \mathbb{1}$.
2. For each $t = 0, \ldots, T-1$ it holds that ρ_{t+1} and σ_{t+1} are density operators and satisfy
$$\begin{aligned} \|\rho_{t+1} - X_{t+1}/\operatorname{Tr}(X_{t+1})\| &< \frac{\varepsilon^2}{2\alpha^2 N^2}, \\ \|\sigma_{t+1} - Y_{t+1}/\operatorname{Tr}(Y_{t+1})\| &< \frac{\varepsilon^2}{4N}, \end{aligned}$$
where X_{t+1} and Y_{t+1} are exactly as defined by the algorithm:
$$\begin{aligned} X_{t+1} &= \exp\left(-\varepsilon\delta \sum_{j=0}^{t} P_\alpha (\Delta_j \otimes \mathbb{1}_{\mathcal{Y}}) P_\alpha / \beta_j \right), \\ Y_{t+1} &= \exp\left(\varepsilon\delta \sum_{j=0}^{t} \operatorname{Tr}_{\mathcal{A}}\left(\Delta_j / \beta_j \right) \right). \end{aligned}$$
(In other words, we do not need to view X_{t+1} and Y_{t+1} as being variables stored by the algorithm, because their only purpose is to define ρ_{t+1} and σ_{t+1}.)
3. The values $\beta_0, \ldots, \beta_{T-1}$ are computed exactly, meaning that $\beta_t = \langle P_\alpha(\Delta_t \otimes \mathbb{1}_{\mathcal{Y}})P_\alpha, \rho_t \rangle$ for $t = 0, \ldots, T-1$. The remaining variables (α, γ, ε, δ, T, and P_α) can obviously be stored exactly as well.

Now, under the assumption (2) it can be shown that

$$\begin{aligned} &\sqrt{\Delta_t} \left(\operatorname{Tr}_{\mathcal{Y}}(P_\alpha \rho_t P_\alpha) - \frac{\gamma}{2} \mathbb{1}_{\mathcal{A}} \otimes \sigma_t \right) \sqrt{\Delta_t} \\ &\quad \geq \Lambda_t \left(\operatorname{Tr}_{\mathcal{Y}}(P_\alpha \rho_t P_\alpha) - \frac{\gamma}{2} \mathbb{1}_{\mathcal{A}} \otimes \sigma_t \right) \Lambda_t - \frac{\varepsilon^2}{4N} \mathbb{1}_{\mathcal{A} \otimes \mathcal{X}}. \qquad (3) \end{aligned}$$

For the case that the algorithm accepts, we use this inequality when performing a similar analysis to the exact case, this time taking

$$\begin{aligned} \rho' &= \frac{\rho}{\gamma + 4\beta + \varepsilon}, \\ \sigma' &= \frac{\gamma\sigma + 4 \operatorname{Tr}_{\mathcal{A}} \left[\sqrt{\Delta} \operatorname{Tr}_{\mathcal{Y}} \left(P_\alpha \rho P_\alpha \right) \sqrt{\Delta} \right] + (\varepsilon/N)\mathbb{1}_{\mathcal{X}}}{\gamma + 4\beta + \varepsilon}, \end{aligned}$$

to conclude that the primal objective value is at least

$$\frac{1}{\gamma + 5\varepsilon} > \frac{5}{8}.$$

(The inequality (3) is slightly stronger than what is needed to obtain this bound, but we will benefit from the stronger form for the case of rejection.)

For the case that the algorithm rejects we again proceed as in the error-free case, except that this time we may make use of the bounds on $\|\rho_{t+1} - X_{t+1}/\operatorname{Tr}(X_{t+1})\|$ to obtain the inequality

$$\operatorname{Tr}(X_{t+1}) \leq \operatorname{Tr}(X_t) \exp(-\varepsilon\delta(1 - \varepsilon)\exp(-\varepsilon))$$

for each $t = 0, \ldots, T-1$, and therefore

$$\begin{aligned} &\lambda_{\min}(P_\alpha (Q \otimes \mathbb{1}_{\mathcal{Y}}) P_\alpha) \\ &\quad \geq (1 + 4\varepsilon) \left((1 - \varepsilon)\exp(-\varepsilon) - \frac{\log(2N^2)}{T\varepsilon\delta} \right) \geq 1. \end{aligned}$$

Thus, dual feasibility still holds for Q in the approximate case. Along similar lines, we apply the assumed bound on $\|\sigma_{t+1} - Y_{t+1}/\operatorname{Tr}(Y_{t+1})\|$ as well as (3) to obtain

$$\operatorname{Tr}(Y_{t+1}) \leq \operatorname{Tr}(Y_t) \exp\left(\frac{2\varepsilon\delta(1 + \varepsilon)\exp(\varepsilon)}{\gamma} \right),$$

so that the dual objective value obtained by Q satisfies

$$\begin{aligned} \frac{1}{2} \|\operatorname{Tr}_{\mathcal{A}}(Q)\| &\leq (1 + 4\varepsilon) \left(\frac{(1 + \varepsilon)\exp(\varepsilon)}{\gamma} + \frac{\log(N)}{2T\varepsilon\delta} \right) \\ &\leq \frac{1 + 8\varepsilon}{\gamma} < \frac{7}{8}. \end{aligned}$$

4.3 An NC implementation of the algorithm

In this subsection we explain how the algorithm above can be implemented by an NC computation. Recall that the class NC may be defined as the class of all functions (including predicates that represent decision problems) computable by logarithmic-space uniform Boolean circuits of polylogarithmic depth.

The matrices stored and processed by our algorithm will have entries with rational real and imaginary parts, and naturally we assume that rational numbers are represented as pairs of integers in binary notation. All other rational numbers processed by the algorithm are represented in a similar way. With these assumptions in place, we first note that elementary matrix operations, including sums, products, tensor products, inverses, and iterated sums and products of matrices, are known to be in NC. There is an extensive literature on this topic, and we refer the reader to the survey of von zur Gathen [18] for more details.

As was mentioned in the previous subsection, the positive eigenspace computation in step 2(a) and the matrix exponential computation in step 2(b) of the algorithm will not be computed exactly in our implementation. Based on the discussion of precision above, however, it will suffice that the following computational problems can be performed by NC computations.

Matrix exponentials

Input:	An $n \times n$ matrix M, a positive rational number η, and an integer k expressed in unary notation (i.e., 1^k).
Promise:	$\|M\| \leq k$.
Output:	An $n \times n$ matrix X such that $\|\exp(M) - X\| < \eta$.

Positive eigenspace projection

Input:	An $n \times n$ Hermitian matrix H and a positive rational number η.
Output:	An $n \times n$ positive semidefinite matrix $\Delta \leq \mathbb{1}$ such that $\|\Delta - \Lambda\| < \eta$, for Λ being the projection operator onto the positive eigenspace of H.

Indeed, the fact that these computations can be performed in NC provides significantly more precision than is needed for our algorithm.

The fact that matrix exponentials can be approximated in NC follows by truncating the series

$$\exp(M) = \mathbb{1} + M + M^2/2 + M^3/6 + \cdots$$

to a number of terms linear in $k + \log(1/\eta)$. (From a numerical point of view this is not necessarily a good way to compute matrix exponentials [37], but it is arguably the simplest way to prove that the stated problem is in NC.) In the case that M is Hermitian it holds that $\exp(M)$ is positive semidefinite, and one may ensure that the approximation X is positive semidefinite as well by taking an odd number of terms in the truncated series for $\exp(M)$.

That positive eigenspace projections can be approximated in NC relies on the fact that roots of integer polynomials can be approximated to very high precision in NC. This was proved first for polynomials having only real roots in [6], and an alternate proof appears in [9]. While the real-root case is sufficient for our needs, we note that the general case, which allows for complex roots, was solved in [38]. It is important to note that distinct roots of integer polynomials can neither be too close to one another nor to zero [34, 12], and that the accuracy with which the roots may be approximated in NC can be taken to be small even relative to this minimal separation. This allows one to determine without error which root approximations correspond to positive roots (in the real-root case).

Given that the characteristic polynomial of a matrix can be computed in NC [13], it follows that an NC algorithm exists for approximating the eigenvalues of a Hermitian matrix to high precision. Given a sufficiently accurate approximation κ to an eigenvalue λ of a Hermitian matrix H, one may obtain a close approximation of the projection Π onto the eigenspace of H corresponding to λ by dividing the matrix

$$(\kappa\mathbb{1} - H)^{-1}$$

by a close approximation to its eigenvalue of maximum absolute value. (One may perturb κ slightly in the event that $\kappa = \lambda$.) By summing the approximate projections onto the eigenspaces corresponding to distinct positive eigenvalues of H, we obtain a close approximation Δ to Λ as claimed.

Once each of the matrix computations required by the algorithm has been implemented as an NC computation, it is straightforward to implement the entire algorithm by an NC computation. We may agree from the start that the real and imaginary parts of each entry of the matrices $\Delta_0, \ldots, \Delta_{T-1}$, $\rho_0, \ldots, \rho_T$ and $\sigma_0, \ldots, \sigma_T$ is to be stored by the algorithm as an integer divided by 2^K, for $K = O(N)$ (or even $K = O(\log N)$ if one prefers), as rounding in this way can only introduce small errors that are within our accuracy requirements. This ensures that the size of the entire computation remains polynomially bounded. As the number of iterations of the algorithm is logarithmic in N, and therefore logarithmic in the size of the input to the algorithm, a composition of the computations described above yields an NC implementation of the algorithm.

5. CONTAINMENT OF QIP IN PSPACE

With the algorithm from the previous section in hand, the containment QIP $\subseteq$ PSPACE may be proved as described in this section.

First, we consider a scaled-up variant of NC, which is the complexity class NC$(poly)$ that consists of all functions computable by polynomial-space uniform families of Boolean circuits having polynomial-depth. (The notation NC(2^{poly}) has also previously been used for this class [11].) For decision problems, it is known that NC$(poly)$ = PSPACE [10]. Thus, it suffices for us to prove QIP $\subseteq$ NC$(poly)$.

A basic property of NC and NC$(poly)$ that we now require is that functions in NC and NC$(poly)$ compose well. Specifically, if F is a function in NC$(poly)$ and G is a function in NC, then the composition $G \circ F$ is also in NC$(poly)$. This follows from the most straightforward way of composing the families of circuits that compute F and G.

Now, assume that A is a promise problem in QIP. Our goal is to prove that $A \in$ NC$(poly)$. As $A \in$ QIP there must exist a quantum interactive proof of the sort described in Section 3 that has perfect completeness and soundness error at most $1/2 - \varepsilon$ for $\varepsilon = 1/64$. (Any other sufficiently small positive constant would do, and in fact one can replace ε with an exponentially small value—but this choice is sufficient for our needs.) We consider a two-step computation as follows:

1. Compute from a given input string x an explicit description of the projections Π_0^0, Π_1^0, Π_0^1, and Π_1^1 that determine the quantum interactive proof system for A that is under consideration.
2. Run an NC implementation of the algorithm described in Section 4 on Π_0^0, Π_1^0, Π_0^1, and Π_1^1.

If it is the case that x is a yes-instance of A, then the optimal value of the semidefinite program described in Section 3 is at least 1, and therefore the algorithm accepts. If x is a no-instance of A, then the optimal value of the semidefinite program is at most $1/2 + \varepsilon + 1/\alpha^2 < 5/8$, and therefore the algorithm rejects.

It remains to observe that the computation can be implemented in NC(*poly*). The first step of this computation can be implemented exactly in NC(*poly*) by computing explicit matrix representations of all of the gates in the quantum circuit specifying the verifier's measurements, followed by elementary matrix computations that yield Π_0^0, Π_1^0, Π_0^1, and Π_1^1. The second step has been described in the previous section. We have that the composition of the two steps is an NC(*poly*) computation, and therefore $A \in$ NC(*poly*) as required.

6. OPEN PROBLEMS

We will conclude this paper by discussing two directions for future research.

The question that is perhaps most clearly raised by this paper is: which semidefinite programs can be solved approximately in parallel (by NC algorithms)? We do not have an answer to this question, and believe it is certainly deserving of further investigation. The fact that the algorithm presented in this paper can be implemented in NC relies heavily on the specific form of the semidefinite program it approximates, and we do not know to what extent the method can be applied to more general classes of semidefinite programs. It is, of course, highly unlikely that all semidefinite programs can be approximated to high accuracy in NC—for even if this were true just for linear programs it would imply NC = P [14, 40, 36].

A second open problem relating to this paper is whether QRG(2) = PSPACE. The class QRG(2) contains all problems having two-turn (i.e., one-round) quantum refereed games, or, in other words, competing-prover quantum interactive proof systems in which the verifier asks each prover a question (in parallel), receives their answers, and makes a decision to accept or reject. The optimal accept/reject probabilities for quantum refereed games can be represented by semidefinite programs [24] and the classical analogue of this class is known to coincide with PSPACE [16].

7. ACKNOWLEDGEMENTS

We thank Xiaodi Wu for helpful discussions. Rahul Jain's research is supported by the internal grants of the Centre for Quantum Technologies, which is funded by the Singapore Ministry of Education and the Singapore National Research Foundation. Zhengfeng Ji acknowledges support from NSF of China (Grant Nos. 60736011 and 60721061); his research at the Perimeter Institute is supported by the Government of Canada through Industry Canada and by the Province of Ontario through the Ministry of Research and Innovation. Sarvagya Upadhyay's research is supported by NSERC, CIFAR, MITACS, QuantumWorks, Industry Canada, Ontario's Ministry of Research and Innovation, and the U.S. ARO. John Watrous's research is supported by NSERC, CIFAR, and QuantumWorks.

8. REFERENCES

[1] S. Arora and B. Barak. *Computational Complexity: A Modern Approach*. Cambridge University Press, 2009.

[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 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.

[4] L. Babai. Trading group theory for randomness. In *Proceedings of the 17th Annual ACM Symposium on Theory of Computing*, pages 421–429, 1985.

[5] L. Babai and S. Moran. Arthur-Merlin games: a randomized proof system, and a hierarchy of complexity classes. *Journal of Computer and System Sciences*, 36(2):254–276, 1988.

[6] M. Ben-Or, E. Feig, D. Kozen, and P. Tiwari. A fast parallel algorithm for determining all roots of a polynomial with real roots. In *Proceedings of the 18th Annual ACM Symposium on Theory of Computing*, pages 340–349, 1986.

[7] M. Ben-Or, S. Goldwasser, J. Kilian, and A. Wigderson. Multi-prover interactive proofs: how to remove intractability assumptions. In *Proceedings of the 20th Annual ACM Symposium on Theory of Computing*, pages 113–131, 1988.

[8] R. Bhatia. *Matrix Analysis*. Springer, 1997.

[9] D. Bini and V. Pan. Computing matrix eigenvalues and polynomial zeros where the output is real. *SIAM Journal on Computing*, 27(4):1099–1115, 1998.

[10] A. Borodin. On relating time and space to size and depth. *SIAM Journal on Computing*, 6:733–744, 1977.

[11] A. Borodin, S. Cook, and N. Pippenger. Parallel computation for well-endowed rings and space-bounded probabilistic machines. *Information and Control*, 58:113–136, 1983.

[12] Y. Bugeaud. *Approximation by Algebraic Numbers*, volume 160 of *Cambridge Tracts in Mathematics*. Cambridge University Press, Cambridge, 2004.

[13] L. Csanky. Fast parallel matrix inversion algorithms. *SIAM Journal on Computing*, 5(4):618–623, 1976.

[14] D. Dobkin, R. Lipton, and S. Reiss. Linear programming is log-space hard for P. *Information Processing Letters*, 8(2):96–97, 1979.

[15] S. Even, A. Selman, and Y. Yacobi. The complexity of promise problems with applications to public-key cryptography. *Information and Control*, 61(2):159–173, 1984.

[16] U. Feige and J. Kilian. Making games short. In *Proceedings of the 29th Annual ACM Symposium on Theory of Computing*, pages 506–516, 1997.

[17] P. Feldman. The optimum prover lies in PSPACE. Manuscript, 1986.

[18] J. von zur Gathen. Parallel linear algebra. In J. Reif, editor, *Synthesis of Parallel Algorithms*, chapter 13. Morgan Kaufmann Publishers, Inc., 1993.

[19] O. Goldreich. On promise problems (a survey in memory of Shimon Even [1935–2004]). Electronic Colloquium on Computational Complexity, Report TR05-018, 2005.

[20] O. Goldreich, S. Micali, and A. Wigderson. Proofs that yield nothing but their validity or all languages in NP have zero-knowledge proof systems. *Journal of the ACM*, 38(1):691–729, 1991.

[21] S. Goldwasser, S. Micali, and C. Rackoff. The knowledge complexity of interactive proof systems. In *Proceedings of the 17th Annual ACM Symposium on Theory of Computing*, pages 291–304, 1985.

[22] S. Goldwasser, S. Micali, and C. Rackoff. The knowledge complexity of interactive proof systems. *SIAM Journal on Computing*, 18(1):186–208, 1989.

[23] S. Goldwasser and M. Sipser. Private coins versus public coins in interactive proof systems. In S. Micali, editor, *Randomness and Computation*, volume 5 of *Advances in Computing Research*, pages 73–90. JAI Press, 1989.

[24] G. Gutoski and J. Watrous. Toward a general theory of quantum games. In *Proceedings of the 39th Annual ACM Symposium on Theory of Computing*, pages 565–574, 2007.

[25] S. Hallgren, A. Kolla, P. Sen, and S. Zhang. Making classical honest verifier zero knowledge protocols secure against quantum attacks. In *Proceedings of the 35th International Colloquium on Automata, Languages and Programming*, volume 5126 of *Lecture Notes in Computer Science*, pages 592–603. Springer, 2008.

[26] R. Jain, S. Upadhyay, and J. Watrous. Two-message quantum interactive proofs are in PSPACE. In *Proceedings of the 50th Annual IEEE Symposium on Foundations of Computer Science*, pages 534–543, 2009.

[27] R. Jain and J. Watrous. Parallel approximation of non-interactive zero-sum quantum games. In *Proceedings of the 24th IEEE Conference on Computational Complexity*, pages 243–253, 2009.

[28] S. Kale. *Efficient algorithms using the multiplicative weights update method*. PhD thesis, Princeton University, 2007.

[29] J. Kempe, H. Kobayashi, K. Matsumoto, and T. Vidick. Using entanglement in quantum multi-prover interactive proofs. *Computational Complexity*, 18(2):273–307, 2009.

[30] A. Kitaev and J. Watrous. Parallelization, amplification, and exponential time simulation of quantum interactive proof system. In *Proceedings of the 32nd Annual ACM Symposium on Theory of Computing*, pages 608–617, 2000.

[31] H. Kobayashi. General properties of quantum zero-knowledge proofs. In *Proceedings of the Fifth IACR Theory of Cryptography Conference*, volume 4948 of *Lecture Notes in Computer Science*, pages 107–124. Springer, 2008.

[32] H. Kobayashi and K. Matsumoto. Quantum multi-prover interactive proof systems with limited prior entanglement. *Journal of Computer and System Sciences*, 66(3):429–450, 2003.

[33] C. Lund, L. Fortnow, H. Karloff, and N. Nisan. Algebraic methods for interactive proof systems. *Journal of the ACM*, 39(4):859–868, 1992.

[34] K. Mahler. *Lectures on Diophantine Approximations*, volume 1. Cushing Malloy, 1961.

[35] C. Marriott and J. Watrous. Quantum Arthur-Merlin games. *Computational Complexity*, 14(2):122–152, 2005.

[36] N. Megiddo. A note on approximate linear programming. *Information Processing Letters*, 42(1):53, 1992.

[37] C. Moler and C. V. Loan. Nineteen dubious ways to compute the exponential of a matrix, twenty-five years later. *SIAM Review*, 45(1):3–49, 2003.

[38] C. A. Neff. Specified precision polynomial root isolation is in NC. *Journal of Computer and System Sciences*, 48(3):429–463, 1994.

[39] M. A. Nielsen and I. L. Chuang. *Quantum Computation and Quantum Information*. Cambridge University Press, 2000.

[40] M. Serna. Approximating linear programming is log-space complete for P. *Information Processing Letters*, 37(4):233–236, 1991.

[41] A. Shamir. IP = PSPACE. *Journal of the ACM*, 39(4):869–877, 1992.

[42] A. Shen. IP = PSPACE: simplified proof. *Journal of the ACM*, 39(4):878–880, 1992.

[43] M. Sipser. *Introduction to the Theory of Computation*. Thomson Course Technology, second edition, 2005.

[44] M. Warmuth and D. Kuzmin. Online variance minimization. In *Proceedings of the 19th Annual Conference on Learning Theory*, volume 4005 of *Lecture Notes in Computer Science*, pages 514–528. Springer, 2006.

[45] J. Watrous. PSPACE has constant-round quantum interactive proof systems. In *Proceedings of the 40th Annual IEEE Symposium on Foundations of Computer Science*, pages 112–119, 1999.

[46] J. Watrous. Zero-knowledge against quantum attacks. *SIAM Journal on Computing*, 39(1):25–58, 2009.