QIP = PSPACE

RAHUL JAIN, National University of Singapore
ZHENGFENG JI, Perimeter Institute for Theoretical Physics and Institute of Software, Chinese Academy of Sciences
SARVAGYA UPADHYAY and JOHN WATROUS, University of Waterloo

This work considers the quantum interactive proof system model of computation, which is the (classical) interactive proof system model's natural quantum computational analogue. An exact characterization of the expressive power of quantum interactive proof systems is obtained: the collection of computational problems having quantum interactive proof systems consists precisely of those problems solvable by deterministic Turing machines that use at most a polynomial amount of space (or, more succinctly, QIP = PSPACE). This characterization is proved through the use of a parallelized form of the matrix multiplicative weights update method, applied to a class of semidefinite programs that captures the computational power of quantum interactive proof systems. One striking implication of this characterization is that quantum computing provides no increase in computational power whatsoever over classical computing in the context of interactive proof systems, for it is well known that the collection of computational problems having classical interactive proof systems coincides with those problems solvable by polynomial-space computations.

Categories and Subject Descriptors: F.1.3 [**Computation by Abstract Devices**]: Complexity Measures and Classes—*Relations among complexity classes*; F.2.1 [**Analysis of Algorithms and Problem Complexity**]: Numerical Algorithms and Problems—*Computations on matrices*

General Terms: Theory

Additional Key Words and Phrases: Interactive proof systems, quantum computation, semidefinite programming, matrix multiplicative weights update method

ACM Reference Format:
Jain, R., Ji, Z., Upadhyay, S., and Watrous, J. 2011. QIP = PSPACE. J. ACM 58, 6, Article 30 (December 2011), 27 pages.
DOI = 10.1145/2049697.2049704 http://doi.acm.org/10.1145/2049697.2049704

The research of R. Jain was 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. The work of Z. Ji was supported by NSF of China (Grant Nos. 60736011 and 60721061). His research at the Perimeter Institute was supported by the Government of Canada through Industry Canada and by the Province of Ontario through the Ministry of Research and Innovation. The research of S. Upadhyay was supported by NSERC, CIFAR, MITACS, QuantumWorks, Industry Canada, Ontario's Ministry of Research and Innovation, and the U.S. ARO. The research of J. Watrous was supported by NSERC, CIFAR, and QuantumWorks.
Authors' addresses: R. Jain, Department of Computer Science, National University of Singapore, S15 #04-01, 3 Science Drive 2, Singapore 117543; email: rahul@comp.nus.edu.sg; Z. Ji, Perimeter Institute for Theoritical Physics, 31 Caroline Street North, Waterloo, Ontario, Canada, N2L 2Y5; email: jizhengfeng@gmail.com; zji@perimeterinstitute.ca; S. Upadhyay and J. Watrous, David R. Cheriton School of Computer Science and Institute for Quantum Computing, University of Waterloo, 200 University Avenue West, Waterloo, ON N2L 3G1, Canada; email: {supadhya, watrous}@cs.uwaterloo.ca.

DOI 10.1145/2049697.2049704 http://doi.acm.org/10.1145/2049697.2049704

1. INTRODUCTION

The notion of a proof has fundamental importance in the theory of computation. Indeed, the foundational work of Church [1936] and Turing [1937] in the 1930s, which produced the first formal models of computation (λ-calculus and Turing machines), was principally motivated by questions concerning proofs in formal logic. The theory of NP-completeness, developed in the 1970s by Cook [1971], Karp [1972] and Levin [1973], provides another example. It is built on the notion of efficient proof verification, and is among the most widely applicable discoveries ever made in the theory of computation.

This article is concerned with the potential advantages offered by *quantum computation* in the setting of proofs, and in particular its advantages when applied to the *interactive proof system* model of computation. Considered by many to be a cornerstone of modern computational complexity theory, the interactive proof system model was first introduced in the mid-1980s, and its quantum computational variant has been an object of study in quantum computing for more than a decade. The main result to be presented herein is that quantum computation does not enhance the expressive power of interactive proof systems: quantum and classical interactive proof systems are equivalent in power, both coinciding with the complexity class PSPACE of problems solvable by deterministic Turing machines that use an amount of space scaling polynomially in the length of the input to the problem. This resolves a fundamental question about the quantum interactive proof system model that has been open since its introduction [Watrous 1999].

1.1. Classical Interactive Proofs

When speaking of proofs, one typically has a traditional notion in mind where, at least in an abstract sense, the proof itself is a string of symbols to be verified by some form of computation. A natural complexity-theoretic abstraction of this notion requires that the proof verification is computationally efficient—it 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.

There are, however, other interesting notions of proofs that extend the traditional notion in various ways. In particular, Babai [1985] and Goldwasser et al. [1985] introduced a computational model that extends the notion of efficient proof verification to an interactive setting and that also makes essential use of *randomness*. (Journal versions of these papers appeared as Babai and Moran [1988] and Goldwasser et al. [1989].) In this model, which is known as the interactive proof system model, a computationally efficient *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 take to be a promise problem [Even et al. 1984; Goldreich 2005] in the interest of generality, is said to have an interactive proof system if there exists a verifier with two properties that reflect the essential meaning of a proof.

(1) *Completeness.* For any *yes*-input string x, there exists a behavior of the prover that, with high probability, causes the verifier to conclude that x is indeed a *yes*-input. This situation is indicated by the verifier outputting 1 (or accept) after interacting with the prover.
(2) *Soundness.* For any *no*-input string x, the verifier will not conclude that x is a *yes*-input, except perhaps with a small probability, regardless of the behavior of the

prover. The verifier outputs 0 (or *reject*) to indicate the prover's failure to convince it that x is a *yes*-input.

Intuitively speaking, the completeness property represents the requirement that true statements can be proved, while the soundness property represents the complementary requirement that false statements cannot be proved. One denotes by IP the collection of decision problems having efficient (i.e., polynomial-time) verifiers that meet these conditions.

It is appropriate to make note of the probabilistic nature of the completeness and soundness properties just described. While one would not expect a formal notion of proof to allow for a nonzero probability of error in the setting of mathematical logic, the interactive proof system model allows for a small, but nevertheless nonzero, error probability. In this way, the model may be seen as describing a *cryptographic* notion of proof, as opposed to one of mathematical logic. Although a nonzero error probability can be eliminated from the completeness condition, it is known that a nonzero soundness error must be permitted if interaction is to provide additional computational power over NP [Furer et al. 1989].

The most fundamental question about the interactive proof system model, from the viewpoint of complexity theory, is: Which decision problems have interactive proof systems? The answer to this question is well known. A problem has an interactive proof system if and only if it is solvable by a deterministic Turing machine running in polynomial space (or, more succinctly, IP = PSPACE). The containment IP $\subseteq$ PSPACE is fairly straightforward to establish, as was first demonstrated by Feldman [1986], using a space-efficient recursive traversal of a *game tree* whose edges represent messages exchanged between the prover and verifier. (Interested readers may find a proof in the textbook of Sipser [1997].) The reverse containment, PSPACE $\subseteq$ IP, was proved by Shamir [1992], based on the work of Lund et al. [1992]. This was a landmark result that established the powerful proof technique of *arithmetization* as a standard tool in computational complexity.

Several variants of interactive proof systems have been studied, including public-coin interactive proof systems [Babai 1985; Babai and Moran 1988; Goldwasser and Sipser 1989], zero-knowledge interactive proof systems [Goldreich et al. 1991; Goldwasser et al. 1985, 1989], multi-prover interactive proof systems [Ben-Or et al. 1988; Babai et al. 1991], and interactive proof systems with competing provers [Feige and Kilian 1997].

1.2. Quantum Computing and Quantum Interactive Proofs

The idea of quantum computation was born in the early 1980s when Feynman [1982] asked a brilliant question: If quantum mechanics is so hard to simulate with a classical computer, why not build a computer based on quantum mechanics to simulate quantum mechanics more directly? Feynman's ideas on the subject led Deutsch [1985] to define the *quantum Turing machine* model and to investigate its computational power. Driven in large part by Shor's discoveries of polynomial-time algorithms for factoring and computing discrete logarithms on a quantum computer [Shor 1997], quantum computation has developed into an active field of study within theoretical computer science and both theoretical and experimental physics.

Large-scale quantum computers do not currently exist, and it is universally agreed that at the very least their realization will be an enormous technological challenge. However, it must be appreciated that quantum mechanics is a remarkably accurate theory that has never been refuted—so scientists are naturally compelled to test the theory of quantum mechanics by attempting to build a quantum computer. Efforts to do this are underway in many laboratories around the world.

Within the theoretical study of quantum computation, it is natural to consider quantum computational variants of interesting classical models, including those based on the notion of proofs. The quantum interactive proof system model, which was first introduced by Watrous [1999], represents a natural quantum computational analogue to the (classical) interactive proof system model. In simple terms, the quantum model allows the verifier and prover in an interactive proof system to perform quantum computations and exchange quantum information, but is otherwise similar to the classical model. In particular, the verifier in a quantum interactive proof system may perform efficient (polynomial-time) quantum computations while the prover can perform quantum computations of arbitrary complexity.

It is important to stress that the potential advantages of quantum computation in the setting of interactive proof systems are not limited to the verifier's ability to perform a potentially wider range of computations. The nature of quantum information is such that it has striking benefits in a variety of information-processing tasks, such as private key exchange [Bennett and Brassard 1984] and distributed computational tasks allowing limited communication [Buhrman et al. 1998; Raz 1999]. A known benefit of quantum computation in the interactive proof system model is that it allows for a major reduction in the number of messages that must be exchanged: quantum interactive proof systems allowing just three messages to be exchanged between the prover and verifier have the full power of those allowing any polynomial number of messages [Kitaev and Watrous 2000]. It is not known if the analogous fact holds for classical interactive proof systems, but it is conjectured not to hold. It would, in particular, imply the collapse of the polynomial-time hierarchy to the second level [Babai and Moran 1988; Goldwasser and Sipser 1989].

Like the classical interactive proof system model, several variants of quantum interactive proof systems have been studied, including those considered by Kobayashi and Matsumoto [2003], Marriott and Watrous [2005], Gutoski and Watrous [2007], Hallgren et al. [2008], Kobayashi [2008], Kempe et al. [2009], and Watrous [2009]. This article relies heavily on a result of Marriott and Watrous [2005] that establishes that quantum interactive proof systems with a very simple form, known as *quantum Arthur-Merlin games*, have the full power of general quantum interactive proof systems. This characterization is described in greater detail later in this article.

The complexity class QIP is defined to be the class of decision problems having efficient quantum verifiers meeting completeness and soundness conditions similar to those described previously for the classical setting. 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. Unlike the subset relation IP $\subseteq$ PSPACE, however, it is not straightforward to prove QIP $\subseteq$ PSPACE. The best upper bound on QIP known prior to the present paper was QIP $\subseteq$ EXP, where EXP denotes the class of problems solvable by deterministic Turing machines running in exponential time. This containment was proved by Kitaev and Watrous [2000] 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 the optimal value of 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. We resolve this problem by proving QIP $\subseteq$ PSPACE, thereby establishing QIP = PSPACE.

1.3. The Matrix Multiplicative Weights Update Method

The key algorithmic technique that allows us to prove QIP $\subseteq$ PSPACE is a method sometimes known as the *matrix multiplicative weights update method*. The ordinary (non-matrix) multiplicative weights method is a framework for algorithm design having its origins in various fields, including learning theory, game theory, and combinatorial optimization. Its matrix variant, as discussed in the survey paper of Arora et al. [2005] and the PhD thesis of Kale [2007], provides an iterative way to efficiently approximate optimal values of certain semidefinite programs [Arora and Kale 2007; Warmuth and Kuzmin 2006].

An important aspect of the matrix multiplicative weights update method from the viewpoint of this article is its *parallelizability* for certain restricted classes of semidefinite programs. Three of us [Jain et al. 2009] previously used an algorithm based on the matrix multiplicative weights update method to prove that QIP(2), the class of problem having 2-message quantum interactive proof systems, is contained in PSPACE. Prior to that work, two of us [Jain and Watrous 2009] used a different algorithm based on the matrix multiplicative weights update method to prove the containment of a different quantum complexity class (namely QRG(1)) in PSPACE.

2. PRELIMINARIES

This section summarizes background material that is required for an understanding of subsequent sections of this article. It is not meant to be comprehensive: it is intended mainly to introduce notations and to highlight certain facts that may be unfamiliar to some readers. A basic understanding of both computational complexity theory and quantum computation is also required, and we refer readers interested in presentations of these topics to Arora and Barak [2009] and Nielsen and Chuang [2000].

2.1. Linear Algebra Basics

Throughout this article, the scripted letters $\mathcal{V}$, $\mathcal{X}$, and $\mathcal{Y}$ denote finite-dimensional complex vector spaces taking the form $\mathbb{C}^\Sigma$ for some finite and nonempty index set Σ. As is typical, we write $\mathbb{C}^N$ rather than $\mathbb{C}^{\{1,\ldots,N\}}$, and we note that the choice to allow index sets besides $\{1,\ldots,N\}$ is purely one of convenience: for any finite and nonempty index set Σ, one can equate $\mathbb{C}^\Sigma$ with $\mathbb{C}^N$ for $N = |\Sigma|$ through any fixed choice of a bijective correspondence between Σ and $\{1,\ldots,N\}$.

On such a space $\mathcal{V} = \mathbb{C}^\Sigma$, one defines a standard inner product as

$$\langle u, v\rangle = \sum_{a\in\Sigma} \overline{u(a)}v(a)$$

for all $u, v \in \mathcal{V}$ (where, in general, $\overline{\alpha}$ denotes the complex conjugate of $\alpha \in \mathbb{C}$). This inner product defines the Euclidean norm

$$\|u\| = \sqrt{\langle u, u\rangle} = \sqrt{\sum_{a\in\Sigma} \big|u(a)\big|^2}.$$

The space of all linear mappings (or *operators*) from $\mathcal{V}$ to itself is denoted $\mathrm{L}(\mathcal{V})$. One may identify elements in this space with matrices whose rows and columns are indexed by Σ (with the notation $A(a,b)$ denoting the entry of a matrix A indexed by the pair (a,b)) in the usual way, and we will freely switch between speaking of operators and

matrices as the context favors. The identity operator on $\mathcal{V}$ is denoted $\mathbb{1}_{\mathcal{V}}$, or simply by $\mathbb{1}$ when $\mathcal{V}$ can safely be taken as implicit. The *trace* of $A \in \mathrm{L}(\mathcal{V})$ is defined as

$$\mathrm{Tr}(A) = \sum_{a \in \Sigma} A(a, a).$$

For each $A \in \mathrm{L}(\mathcal{V})$ one defines the *adjoint* operator $A^* \in \mathrm{L}(\mathcal{V})$ as the unique operator satisfying $\langle u, Av \rangle = \langle A^* u, v \rangle$ for all $u, v \in \mathcal{V}$. As a matrix, A^* is obtained by taking the conjugate transpose of A:

$$A^*(a, b) = \overline{A(b, a)}.$$

It is convenient to define an inner product on $\mathrm{L}(\mathcal{V})$ as

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

For an operator $A \in \mathrm{L}(\mathcal{V})$, a complex number $\lambda \in \mathbb{C}$ is said to be an *eigenvalue* of A if there exists a nonzero vector $u \in \mathcal{V}$ such that $Au = \lambda u$. Any such vector u is said to be an *eigenvector* of A whose associated eigenvalue is λ.

The *spectral norm* of an operator $A \in \mathrm{L}(\mathcal{V})$ is defined as

$$\|A\| = \max\{\|Au\| \,:\, u \in \mathcal{V},\ \|u\| = 1\},$$

where $\|Au\|$ and $\|u\|$ refer to the Euclidean norm on $\mathcal{V}$. The spectral norm is submultiplicative: $\|AB\| \leq \|A\| \, \|B\|$ for all operators $A, B \in \mathrm{L}(\mathcal{V})$.

We write $\mathrm{Herm}(\mathcal{V})$ to denote the subset of $\mathrm{L}(\mathcal{V})$ consisting of all Hermitian, or self-adjoint, operators:

$$\mathrm{Herm}(\mathcal{V}) = \left\{ A \in \mathrm{L}(\mathcal{V}) \,:\, A = A^* \right\}.$$

It holds that $\langle A, B \rangle$ is a real number for all Hermitian operators $A, B \in \mathrm{Herm}(\mathcal{V})$, and every eigenvalue of a Hermitian operator is necessarily a real number as well. The notations $\lambda_{\min}(A)$ and $\lambda_{\max}(A)$ refer, respectively, to the smallest and largest eigenvalues of a Hermitian operator A. When we refer to the *negative eigenspace* of a Hermitian operator $A \in \mathrm{Herm}(\mathcal{V})$, we are referring to the subspace of $\mathcal{V}$ spanned by every vector $u \in \mathcal{V}$ that is an eigenvector of A having an associated eigenvalue that is negative.

An operator $P \in \mathrm{L}(\mathcal{V})$ is *positive semidefinite* if and only if $P = A^* A$ for some choice of $A \in \mathrm{L}(\mathcal{V})$. It holds that $P \in \mathrm{L}(\mathcal{V})$ is positive semidefinite if and only if (i) $P \in \mathrm{Herm}(\mathcal{V})$ and (ii) every eigenvalue of P is nonnegative. The set of all positive semidefinite operators on $\mathcal{V}$ will be denoted $\mathrm{Pos}(\mathcal{V})$, so that

$$\mathrm{Pos}(\mathcal{V}) = \left\{ A^* A \,:\, A \in \mathrm{L}(\mathcal{V}) \right\}.$$

We note that $\langle P, Q \rangle$ is a nonnegative real number for all positive semidefinite operators $P, Q \in \mathrm{Pos}(\mathcal{V})$. For Hermitian operators A and B, the notations $A \leq B$ and $B \geq A$ mean that $B - A$ is positive semidefinite. A positive semidefinite operator $\Pi \in \mathrm{Pos}(\mathcal{V})$ is a *projection* operator if and only if $\Pi^2 = \Pi$, or equivalently if and only if all of its eigenvalues are either 0 or 1.

Finally, an operator $U \in \mathrm{L}(\mathcal{V})$ is *unitary* if and only if $UU^* = \mathbb{1}_{\mathcal{V}} = U^* U$.

2.2. Quantum Information Basics

When we speak of a *register* in this article, we mean a finite and nonempty sequence of qubits that we view as a single entity and to which we assign a name. The names X and Y, in sans serif font, will be used to refer to registers. When a register X consists of m qubits we say that X is an m-qubit register, and in this case it has an associated vector space $\mathcal{X} = \mathbb{C}^{\{0,1\}^m}$. (We take $\mathcal{Y}$ to be the vector space associated with the register Y in a similar way.)

The possible *quantum states* (or simply *states*) of a register X correspond to the set $\mathrm{D}(\mathcal{X})$ of *density operators* on $\mathcal{X}$, which are the positive semidefinite operators having unit trace:

$$\mathrm{D}(\mathcal{X}) = \left\{\rho \in \mathrm{Pos}(\mathcal{X}) \,:\, \mathrm{Tr}(\rho) = 1\right\}.$$

It is traditional in quantum physics to use lower-case Greek letters, often ρ, σ and ξ, to represent density operators.

A *measurement* on a register X is a finite collection of positive semidefinite operators $\{P_b \,:\, b \in \Gamma\} \subseteq \mathrm{Pos}(\mathcal{X})$ 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 operator.

The space associated with a pair of registers is given by the *tensor product* of the spaces associated with the individual registers. In particular, for registers X and Y having associated spaces $\mathcal{X}$ and $\mathcal{Y}$, we have that the space associated with the pair (X, Y) is $\mathcal{X} \otimes \mathcal{Y}$. If X and Y are registers *independently* prepared in the quantum states represented by $\sigma \in \mathrm{D}(\mathcal{X})$ and $\xi \in \mathrm{D}(\mathcal{Y})$, then the state of the pair (X, Y) is understood to coincide with the density operator $\sigma \otimes \xi \in \mathrm{D}(\mathcal{X} \otimes \mathcal{Y})$. Of course, not every state of a pair of registers (X, Y) can be expressed as a tensor product in this way, representing the fact that X and Y may be correlated. *Entanglement* is a form of correlation representing one of the most important and distinguishing features of quantum information [Horodecki et al. 2009].

For every state of a pair of registers (X, Y), whether correlated or not, there is a uniquely determined *reduced state* of the register X that, in essence, would describe the state of X if Y were to be destroyed or lost. This is analogous to the marginal probability distribution of X in the probabilistic case. In mathematical terms the reduced state of X is defined by the *partial trace* over $\mathcal{Y}$. This is the unique linear mapping $\mathrm{Tr}_{\mathcal{Y}}$ from $\mathrm{L}(\mathcal{X} \otimes \mathcal{Y})$ to $\mathrm{L}(\mathcal{X})$ that satisfies the equation

$$\mathrm{Tr}_{\mathcal{Y}}(A \otimes B) = \mathrm{Tr}(B)A$$

for every $A \in \mathrm{L}(\mathcal{X})$ and $B \in \mathrm{L}(\mathcal{Y})$. We note that

$$\left\langle A, \mathrm{Tr}_{\mathcal{Y}}(B)\right\rangle = \langle A \otimes \mathbb{1}_{\mathcal{Y}}, B\rangle$$

for all choices of $A \in \mathrm{L}(\mathcal{X})$ and $B \in \mathrm{L}(\mathcal{X} \otimes \mathcal{Y})$, which is an identity we will make use of several times.

One additional component of quantum information theory that has not been mentioned yet is the description of *transformations* of quantum registers according to physical processes (such as computations). We will only need to consider *unitary* transformations in this article: if a register X is in a state $\rho \in \mathrm{D}(\mathcal{X})$, and a transformation described by a unitary operator $U \in \mathrm{L}(\mathcal{X})$ is performed on this register, its state becomes $U\rho U^*$.

2.3. A Lemma Underlying the Matrix Multiplicative Weights Update Method

The main algorithm, and its analysis, that we present later in the paper makes use of a lemma to be presented in this section. The lemma is associated with the matrix multiplicative weights update method, and may be found in Kale [2007] (for instance). A proof is included here for the sake of completeness.

We begin with the definition of the exponential function of an operator. For every operator $A \in \mathrm{L}(\mathcal{V})$, the *exponential* of A is defined as

$$\exp(A) = \sum_{n=0}^{\infty} \frac{A^n}{n!}.$$

Two properties of operator exponentials will be required to prove the main lemma of this subsection. One property is the *Golden-Thompson inequality*, which states that

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

for any choice of Hermitian operators $A, B \in \mathrm{Herm}(\mathcal{V})$. A proof of the Golden-Thompson inequality may be found in Section IX.3 of Bhatia [1997]. A second property of operator exponentials that we will require is stated by the following lemma.

LEMMA 1. *Let $R \in \mathrm{Pos}(\mathcal{V})$ be a positive semidefinite operator satisfying $R \leq \mathbb{1}$. For every real number η, it holds that $\exp(\eta R) \leq \mathbb{1} + \eta \exp(\eta) R$.*

PROOF. It is sufficient to prove the inequality for R replaced by a scalar $\lambda \in [0,1]$, for then the operator inequality follows by considering a spectral decomposition of R. If $\lambda = 0$ the inequality is immediate, so assume $0 < \lambda \leq 1$. 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 inequality follows. □

We are now prepared to state and prove the main lemma of this section.

LEMMA 2. *Let T be a positive integer, let $M^{(0)}, \ldots, M^{(T-1)} \in \mathrm{Pos}(\mathcal{V})$ be operators on a space $\mathcal{V} = \mathbb{C}^N$ that satisfy $0 \leq M^{(t)} \leq \mathbb{1}$ for each $t \in \{0, \ldots, T-1\}$, and let $\varepsilon > 0$. The following two facts hold:*

(1) *For*

$$X^{(0)} = \mathbb{1}_{\mathcal{V}}, \quad X^{(t+1)} = \exp\left(-\varepsilon \sum_{j=0}^{t} M^{(j)}\right), \quad \textit{and} \quad \rho^{(t)} = \frac{X^{(t)}}{\mathrm{Tr}\left(X^{(t)}\right)}$$

for each $t \in \{0, \ldots, T-1\}$, one has

$$\lambda_{\min}\left(\sum_{t=0}^{T-1} M^{(t)}\right) \geq \exp(-\varepsilon) \sum_{t=0}^{T-1} \left\langle \rho^{(t)}, M^{(t)} \right\rangle - \frac{\ln(N)}{\varepsilon}. \tag{1}$$

(2) *For*

$$Y^{(0)} = \mathbb{1}_{\mathcal{V}}, \quad Y^{(t+1)} = \exp\left(\varepsilon \sum_{j=0}^{t} M^{(j)}\right), \quad \textit{and} \quad \sigma^{(t)} = \frac{Y^{(t)}}{\mathrm{Tr}\left(Y^{(t)}\right)}$$

for each $t \in \{0, \ldots, T-1\}$, one has

$$\lambda_{\max}\left(\sum_{t=0}^{T-1} M^{(t)}\right) \leq \exp(\varepsilon) \sum_{t=0}^{T-1} \left\langle \sigma^{(t)}, M^{(t)} \right\rangle + \frac{\ln(N)}{\varepsilon}. \tag{2}$$

PROOF. Let us first prove the first fact stated in the lemma. Consider any choice of $t \in \{0, \ldots, T-1\}$. By the Golden-Thompson inequality it holds that

$$\mathrm{Tr}\left(X^{(t+1)}\right) \leq \mathrm{Tr}\left(X^{(t)} \exp\left(-\varepsilon M^{(t)}\right)\right). \tag{3}$$

By Lemma 1, one has

$$\exp\left(-\varepsilon M^{(t)}\right) \leq \mathbb{1} - \varepsilon \exp(-\varepsilon) M^{(t)}. \tag{4}$$

Given that $X^{(t)}$ is positive semidefinite, Eq. (3) and (4) imply

$$\begin{aligned}\mathrm{Tr}\left(X^{(t+1)}\right) &\leq \mathrm{Tr}\left(X^{(t)} - \varepsilon \exp(-\varepsilon) X^{(t)} M^{(t)}\right) \\ &= \mathrm{Tr}\left(X^{(t)}\right)\left(1 - \varepsilon \exp(-\varepsilon)\left\langle \rho^{(t)}, M^{(t)}\right\rangle\right).\end{aligned}$$

Making use of the fact that $1 + z \leq \exp(z)$ for every real number z, one obtains

$$\mathrm{Tr}\left(X^{(t+1)}\right) \leq \mathrm{Tr}\left(X^{(t)}\right) \exp\left(-\varepsilon \exp(-\varepsilon)\left\langle \rho^{(t)}, M^{(t)}\right\rangle\right). \tag{5}$$

Iteratively applying the inequality (5) to $t = T-1, \ldots, 0$, and using the observation that $\mathrm{Tr}(X^{(0)}) = N$, yields

$$\mathrm{Tr}\left(X^{(T)}\right) \leq N \exp\left(-\varepsilon \exp(-\varepsilon) \sum_{t=0}^{T-1} \left\langle \rho^{(t)}, M^{(t)}\right\rangle\right). \tag{6}$$

Next, using the fact that $\mathrm{Tr}(\exp(-H)) \geq \exp(-\lambda_{\min}(H))$ for every Hermitian operator H, one has that

$$\mathrm{Tr}\left(X^{(T)}\right) \geq \exp\left(-\varepsilon \lambda_{\min}\left(\sum_{t=0}^{T-1} M^{(t)}\right)\right). \tag{7}$$

Combining the inequalities (6) and (7) yields (1) as claimed.

The second fact stated in the lemma is proved in much the same way as the first, except that signs and inequalities are reversed in the most straightforward manner. More specifically, one has

$$\mathrm{Tr}\left(Y^{(t+1)}\right) \leq \mathrm{Tr}\left(Y^{(t)} \exp\left(\varepsilon M^{(t)}\right)\right)$$

for each $t \in \{0, \ldots, T-1\}$ by the Golden-Thompson inequality. By Lemma 1, it holds that $\exp\left(\varepsilon M^{(t)}\right) \leq \mathbb{1} + \varepsilon \exp(\varepsilon) M^{(t)}$, so that

$$\mathrm{Tr}\left(Y^{(t+1)}\right) \leq \mathrm{Tr}\left(Y^{(t)}\right) \exp\left(\varepsilon \exp(\varepsilon)\left\langle \sigma^{(t)}, M^{(t)}\right\rangle\right),$$

and therefore

$$\mathrm{Tr}\left(Y^{(T)}\right) \leq N \exp\left(\varepsilon \exp(\varepsilon) \sum_{t=0}^{T-1} \left\langle \sigma^{(t)}, M^{(t)}\right\rangle\right).$$

Combining this inequality with the inequality

$$\mathrm{Tr}\left(Y^{(T)}\right) \geq \exp\left(\varepsilon \lambda_{\max}\left(\sum_{t=0}^{T-1} M^{(t)}\right)\right)$$

yields (2). This completes the proof. □

We note that the two facts stated in this lemma may be combined into a single inequality, as is done in Kale [2007]. For our purposes, however, it is natural to separate the two statements as we have done.

2.4. Quantum Interactive Proof Systems and Single-Coin Arthur-Merlin Games

Taken in its most general form, the quantum interactive proof system model can allow for complicated interactions between a prover and verifier involving the exchange of quantum messages over the course of many rounds. By the nature of quantum information, such an interaction cannot generally be described by the sort of game tree that describes a classical interaction: the possibility of entanglement among the prover, verifier and message registers prohibits this.

It is, however, always possible to transform a given quantum interactive proof system into one with a conceptually simple form by following the method presented in Section 5 of Marriott and Watrous [2005]. A polynomial p, independent of the input string, is fixed, and the input to the problem under consideration is denoted by x. The verifier expects that the messages sent to it by the prover will be contained in m-qubit registers for $m = p(n)$, where $n = |x|$ is the input length. (Marriott and Watrous did not restrict the prover's messages to have a common length, and we do not really require this restriction either—but given that the restriction can easily be made without altering the power of the model, we will assume it in the interest of simplicity.) The transformed proof system has the following form.

(1) The prover begins by sending the verifier an m-qubit register X. Upon receiving X, the verifier sets it aside without interacting with it.
(2) The verifier chooses a bit $a \in \{0, 1\}$, uniformly at random, and sends a copy of a to the prover.
(3) The prover sends the verifier a second m-qubit register Y.
(4) The verifier measures the pair (X, Y) with respect to one of two binary-outcome measurements: $\{\Pi_0^0, \Pi_1^0\}$ in case $a = 0$ and $\{\Pi_0^1, \Pi_1^1\}$ in case $a = 1$. The measurement outcome is interpreted as the verifier's output: as usual, 1 means the proof is accepted, 0 means it is rejected.

The verifier's measurements $\{\Pi_0^0, \Pi_1^0\}$ and $\{\Pi_0^1, \Pi_1^1\}$ naturally depend on the input string x to the problem under consideration. In accordance with the definition of the quantum interactive proof system model, these measurements must be efficiently implementable by a quantum computation to be performed by the verifier.

In greater detail, we may assume that the verifier performs its measurements by first applying a polynomial number $q(n)$ of unitary operations $U_1, \ldots, U_{q(n)}$ to the $2m$ qubits comprising the pair (X, Y), followed by a measurement of all of these qubits with respect to the standard basis. Each of the operations $U_1, \ldots, U_{q(n)}$ corresponds to either a Toffoli gate, a Hadamard gate, or a $\pi/2$-phase shift (or i-phase shift) gate, and the specification of this sequence of gates is given by a polynomial-time computable function of the input x. The final binary-valued output of the verifier's measurement is obtained by a simple computation on the result obtained from the standard-basis measurement: for the first measurement $\{\Pi_0^0, \Pi_1^0\}$ we may take the output to be 1 if and only if the standard-basis measurement gives an outcome where the first m bits are 0, and for the second measurement $\{\Pi_0^1, \Pi_1^1\}$ we may take the output to be 1 if and only if the standard-basis measurement gives an outcome where the first bit is 1. (Any other polynomial-time computable predicate would be equivalent in terms of our proof. We have only mentioned these particular predicates for concreteness, and because they are the ones required by Marriott and Watrous.) We note that $\{\Pi_0^0, \Pi_1^0\}$ and $\{\Pi_0^1, \Pi_1^1\}$ are projective measurements, and although our proof could be adapted to

more general measurements without significant complications, it will simplify matters for us to make use of this fact.

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 [Babai 1985; Babai and Moran 1988]). By making use of the results of Kitaev and Watrous [2000], Marriott and Watrous [2005] proved that every problem $A \in \mathrm{QIP}$ has a quantum interactive proof system of the preceding form, where the completeness is perfect and the soundness error is bounded by $1/2 + \varepsilon$ for any desired constant $\varepsilon > 0$. In other words, the verifier's measurements can be forced to output 1 with certainty when the input is a *yes*-input, while the maximum probability for the output 1 is at most $1/2 + \varepsilon$ (for any fixed choice of a constant $\varepsilon > 0$) when the input is a *no*-input.

A common question about quantum interactive proof systems having the above form is this: Why cannot the prover simply prepare three registers X, Y_0, and Y_1, send all three to the verifier in a single message, and allow the verifier to measure $(\mathsf{X}, \mathsf{Y}_0)$ or $(\mathsf{X}, \mathsf{Y}_1)$ depending on its choice of the random bit a? This would seem to eliminate the need for interaction, as the verifier would never send anything to the prover.

The reason is that entanglement prevents this from working: in order for the two measurements to both result in the output 1 with high probability, the register pairs $(\mathsf{X}, \mathsf{Y}_0)$ and $(\mathsf{X}, \mathsf{Y}_1)$ would generally need to simultaneously be in highly entangled states (at least for nontrivial proof systems). One of the curious features of entanglement, however, is that any single quantum register is constrained with respect to the degree of entanglement it may simultaneously share with two or more other registers. (This phenomenon is often called the *monogamy of entanglement* [Terhal 2004].) In general, the only way for the prover to cause the verifier to output 1 is to prepare X in a highly entangled state with a register of its own, and then use this entanglement to prepare Y once the random bit a has been received.

2.5. Semidefinite Programs for Single-Coin Quantum Arthur-Merlin Games

There are a variety of strategies that a prover could potentially employ in an attempt to cause a verifier to output 1 in a single-coin quantum Arthur-Merlin game, as discussed above; but they can all be accounted for by considering the possible states of the pair (X, Y) that the verifier measures, conditioned on the two possible values of the random bit a. That is, for any two density operators $\rho_0, \rho_1 \in \mathrm{D}(\mathcal{X} \otimes \mathcal{Y})$, one may consider whether it is possible for the prover to follow a strategy that will leave the verifier with the state ρ_0 for the pair (X, Y) when the random bit takes the value $a = 0$, and with the state ρ_1 when $a = 1$. The set of all such choices for ρ_0 and ρ_1 is simple to characterize: it is precisely the set of all pairs (ρ_0, ρ_1) for which

$$\mathrm{Tr}_{\mathcal{Y}}(\rho_0) = \mathrm{Tr}_{\mathcal{Y}}(\rho_1). \tag{8}$$

The necessity of this condition follows immediately from the fact that the prover cannot touch X at any point after learning a, while its sufficiency follows from the well-known principle in quantum information theory sometimes called the *unitary equivalence of purifications*. It follows that the maximum probability for a prover to cause the verifier to output 1 in the type of proof system just described is the maximum of the quantity

$$\frac{1}{2}\left\langle \Pi_1^0, \rho_0 \right\rangle + \frac{1}{2}\left\langle \Pi_1^1, \rho_1 \right\rangle, \tag{9}$$

subject to the condition that $\rho_0, \rho_1 \in \mathrm{D}(\mathcal{X} \otimes \mathcal{Y})$ satisfy (8).

The problem of maximizing the preceding quantity (9), subject to the constraint (8), is an instance of a *semidefinite programming problem*, and the main result of this article is obtained through an algorithmic method for approximating the optimal solution

to it. In order to make this optimization problem more amenable to our method, we must modify it in the following way.

First, consider the problem of maximizing the quantity

$$\frac{1}{2}\langle \Pi_1^0, Q_0 \rangle + \frac{1}{2}\langle \Pi_1^1, Q_1 \rangle,$$

over all operators $Q_0, Q_1 \in \mathrm{Pos}(\mathcal{X} \otimes \mathcal{Y})$ such that $\mathrm{Tr}_{\mathcal{Y}}(Q_0) \leq \xi$ and $\mathrm{Tr}_{\mathcal{Y}}(Q_1) \leq \xi$ for some choice of a density operator $\xi \in \mathrm{D}(\mathcal{X})$. It is not difficult to see that this problem has the same maximum value as the original problem.

Next, for a positive number α (which we will set as $\alpha = 4$ shortly), we define two operators:

$$P_0 = \Pi_1^0 + \alpha \Pi_0^0 \quad \text{and} \quad P_1 = \Pi_1^1 + \alpha \Pi_0^1.$$

These operators are positive semidefinite and invertible, with inverses given by

$$P_0^{-1} = \Pi_1^0 + \frac{1}{\alpha}\Pi_0^0 \quad \text{and} \quad P_1^{-1} = \Pi_1^1 + \frac{1}{\alpha}\Pi_0^1.$$

It holds that the maximum value of the quantity

$$\frac{1}{2}\left\langle P_0^{-2}, Q_0 \right\rangle + \frac{1}{2}\left\langle P_1^{-2}, Q_1 \right\rangle,$$

subject to the same constraints on Q_0 and Q_1 as before, is at least the original optimal value and at most the original optimal value plus $1/\alpha^2$.

Finally, given that P_0 and P_1 are Hermitian and invertible, we have that $P_0 Q_0 P_0$ ranges over all positive semidefinite operators on $\mathcal{X} \otimes \mathcal{Y}$ as Q_0 does, and likewise for $P_1 Q_1 P_1$ and Q_1. Thus, by a simple change-of-variables, the problem described above is equivalent to maximizing the quantity

$$\mathrm{Tr}\left(\frac{1}{2}Q_0 + \frac{1}{2}Q_1\right)$$

over all $Q_0, Q_1 \in \mathrm{Pos}(\mathcal{X} \otimes \mathcal{Y})$ subject to the constraints that $\mathrm{Tr}_{\mathcal{Y}}(P_0 Q_0 P_0) \leq \xi$ and $\mathrm{Tr}_{\mathcal{Y}}(P_1 Q_1 P_1) \leq \xi$ for some choice of $\xi \in \mathrm{D}(\mathcal{X})$.

To aid in the exposition of the later sections of this article, the semidefinite programming problem just described, together with its dual problem formulation, is presented in the following definition.

Definition 3. For positive semidefinite operators $P_0, P_1 \in \mathrm{Pos}(\mathcal{X} \otimes \mathcal{Y})$, we define the semidefinite program $\mathrm{SDP}(P_0, P_1)$ to consist of the following primal and dual optimization problems:

Primal problem

$$\begin{aligned} \text{maximize:} \quad & \mathrm{Tr}\left(\frac{1}{2}Q_0 + \frac{1}{2}Q_1\right) \\ \text{subject to:} \quad & \mathrm{Tr}_{\mathcal{Y}}(P_0 Q_0 P_0) \leq \xi, \\ & \mathrm{Tr}_{\mathcal{Y}}(P_1 Q_1 P_1) \leq \xi, \\ & Q_0, Q_1 \in \mathrm{Pos}(\mathcal{X} \otimes \mathcal{Y}), \\ & \xi \in \mathrm{D}(\mathcal{X}). \end{aligned}$$

Dual problem

$$\begin{aligned} \text{minimize:} \quad & \left\| \frac{1}{2}R_0 + \frac{1}{2}R_1 \right\| \\ \text{subject to:} \quad & P_0(R_0 \otimes \mathbb{1}_{\mathcal{Y}})P_0 \geq \mathbb{1}_{\mathcal{X}} \otimes \mathbb{1}_{\mathcal{Y}}, \\ & P_1(R_1 \otimes \mathbb{1}_{\mathcal{Y}})P_1 \geq \mathbb{1}_{\mathcal{X}} \otimes \mathbb{1}_{\mathcal{Y}}, \\ & R_0, R_1 \in \mathrm{Pos}(\mathcal{X}). \end{aligned}$$

It is not necessary for this article that readers have familiarity with semidefinite programming duality, and an understanding of how the dual problem above is obtained from the primal problem is not required. All that is necessary is to observe

that *weak duality* holds, which means that every value obtainable for the primal problem is bounded above by every value obtainable for the dual problem. This can be verified directly as follows. Suppose that $Q_0, Q_1 \in \mathrm{Pos}(\mathcal{X} \otimes \mathcal{Y})$ and $\xi \in \mathrm{D}(\mathcal{X})$ satisfy the primal problem constraints $\mathrm{Tr}_{\mathcal{Y}}(P_0 Q_0 P_0) \leq \xi$ and $\mathrm{Tr}_{\mathcal{Y}}(P_1 Q_1 P_1) \leq \xi$, and $R_0, R_1 \in \mathrm{Pos}(\mathcal{X})$ satisfy the dual problem constraints $P_0(R_0 \otimes \mathbb{1}_{\mathcal{Y}})P_0 \geq \mathbb{1}_{\mathcal{X}} \otimes \mathbb{1}_{\mathcal{Y}}$ and $P_1(R_1 \otimes \mathbb{1}_{\mathcal{Y}})P_1 \geq \mathbb{1}_{\mathcal{X}} \otimes \mathbb{1}_{\mathcal{Y}}$. One then has

$$\begin{aligned}
\mathrm{Tr}\left(\frac{1}{2}Q_0 + \frac{1}{2}Q_1\right) &\leq \frac{1}{2}\left\langle P_0(R_0 \otimes \mathbb{1}_{\mathcal{Y}})P_0,\, Q_0\right\rangle + \frac{1}{2}\left\langle P_1(R_1 \otimes \mathbb{1}_{\mathcal{Y}})P_1,\, Q_1\right\rangle \\
&= \frac{1}{2}\left\langle R_0, \mathrm{Tr}_{\mathcal{Y}}(P_0 Q_0 P_0)\right\rangle + \frac{1}{2}\left\langle R_1, \mathrm{Tr}_{\mathcal{Y}}(P_1 Q_1 P_1)\right\rangle \leq \frac{1}{2}\left\langle R_0, \xi\right\rangle + \frac{1}{2}\left\langle R_1, \xi\right\rangle \\
&= \left\langle \frac{1}{2}R_0 + \frac{1}{2}R_1, \xi \right\rangle \leq \left\| \frac{1}{2}R_0 + \frac{1}{2}R_1 \right\|.
\end{aligned}$$

3. PSPACE, BOUNDED-DEPTH CIRCUITS, AND THE MAIN PROOF OVERVIEW

At the heart of our proof of the containment QIP $\subseteq$ PSPACE is a parallel algorithm for approximately solving semidefinite programs of the special form described in the previous section. The purpose of the present section is to specify the precise requirements of this algorithm, and why its existence implies QIP $\subseteq$ PSPACE. The algorithm itself is presented in the section following this one.

Suppose that $A = (A_{\mathrm{yes}}, A_{\mathrm{no}})$ is a promise problem in QIP. Our goal is to prove $A \in$ PSPACE, or, equivalently, that there exists a polynomial-space algorithm for A. While it is evidently possible to describe such an algorithm directly, we find that it is more natural from an algorithmic perspective to translate the task at hand into one involving families of bounded-depth Boolean circuits. This translation is possible using a classic result in circuit complexity due to Borodin [1977].

Recall that NC is defined as the class of all functions (including predicates that represent decision problems) computable by families of logarithmic-space uniform Boolean circuits of polylogarithmic depth. The requirement that such a family is logarithmic-space uniform implies that its circuits are polynomial in size, and therefore represent polynomial-time computations. That these circuits have polylogarithmic depth represents an abstraction of massive parallelizability. Many interesting computational tasks are known to correspond to NC computations, including a wide range of matrix computations.

We also consider a "scaled-up" variant of NC, to be denoted 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 [Borodin et al. 1983].) A family of circuits meeting these requirements could potentially have exponential size, and therefore does not necessarily represent an efficient computation. However, the polynomial bound on the depth of these circuits does represent a significant computational restriction. In particular, restricting our attention to decision problems, we have

$$\mathrm{NC}(poly) = \mathrm{PSPACE}.$$

This equality follows from a more general result of Borodin [1977] that was referred to previously. The primary appeal of this reformulation is that it allows one to make use of extensive work on parallel algorithms for performing various computational tasks when designing PSPACE algorithms.

Given that NC($poly$) = PSPACE, we may consider that the task at hand is to prove $A \in$ NC($poly$). When doing this, we will make use of a property of NC and NC($poly$), which is that functions in these classes compose well. Specifically, if F is a function in

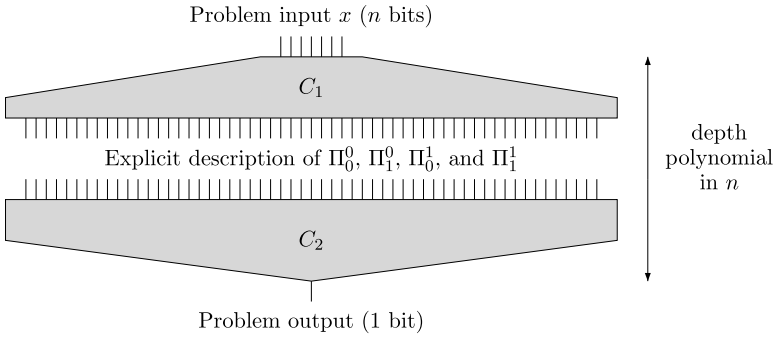

Fig. 1. An illustration of the two-stage process for solving any problem $A \in$ QIP by bounded-depth Boolean circuits. The circuit C_1 transforms the input string into an exponentially large description of the verifier's measurements $\{\Pi_0^0, \Pi_1^0\}$ and $\{\Pi_0^1, \Pi_1^1\}$, and the circuit C_2 implements a parallel algorithm for approximating the optimal value of the associated semidefinite program.

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.

The assumption that $A \in$ QIP implies that there exists a quantum interactive proof system for A with the special structure described in Section 2.4. In particular, we have that the verifier's actions in such a proof system on a fixed input string x are described by projection operators Π_0^0, Π_1^0, Π_0^1, and Π_1^1, and we make the following assumptions on the completeness and soundness of this proof system.

Completeness. If it holds that $x \in A_{\text{yes}}$, then there exists a strategy for the prover that causes the verifier to output 1 (accept) with probability 1. It follows that the optimal value of the semidefinite program $\text{SDP}(P_0, P_1)$, for $P_0 = \Pi_1^0 + \alpha\Pi_0^0$, $P_1 = \Pi_1^1 + \alpha\Pi_0^1$, and $\alpha = 4$, is at least 1.

Soundness. If it holds that $x \in A_{\text{no}}$, then every prover strategy causes the verifier to output 0 (reject) with probability at least $1/2 - \varepsilon$, for $\varepsilon = 1/64$. It follows that the optimal value of the semidefinite program $\text{SDP}(P_0, P_1)$, for $P_0 = \Pi_1^0 + \alpha\Pi_0^0$, $P_1 = \Pi_1^1 + \alpha\Pi_0^1$, and $\alpha = 4$, is at most $1/2 + \varepsilon + 1/\alpha^2 < 5/8$.

(Any other sufficiently small positive constant in place of $\varepsilon = 1/64$ would do. One can, in fact, assume that the probability $1/2 - \varepsilon$ in the soundness condition is replaced by a quantity exponentially close to $1/2$, as a function of n, but such an assumption does not offer any advantages with respect to our analysis.)

With these assumptions in mind we consider a two-stage computation, illustrated in Figure 1, 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. This computation, which is represented by the circuit C_1 in Figure 1, is to be performed in NC(*poly*).
(2) Run an NC implementation of the algorithm described below in Section 4 on Π_0^0, Π_1^0, Π_0^1, and Π_1^1. This computation is represented by the circuit C_2 in Figure 1, and must accept (or output 1) if the optimal value of the semidefinite program $\text{SDP}(P_0, P_1)$ is at least 1, and reject (or output 0) if the optimal value is less than 5/8.

The first stage of this process turns out to be fairly straightforward, using elementary facts about quantum computations and bounded-depth circuits. Making use of the assumption that the verifier's measurements are efficiently implementable (i.e., representable by polynomial-time uniform families of quantum circuits), this stage can be computed exactly in NC(*poly*). This may be done by computing explicit matrix representations of all of the gates in the quantum circuits 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.

In greater detail, suppose that $\{\Pi_0, \Pi_1\}$ is one of the verifier's two measurements. Following the discussion in the previous section, this measurement takes the form

$$\Pi_0 = U_1^* \cdots U_{q(n)}^* \Lambda_0 U_{q(n)} \cdots U_1 \quad \text{and} \quad \Pi_1 = U_1^* \cdots U_{q(n)}^* \Lambda_1 U_{q(n)} \cdots U_1$$

for $U_1, \ldots, U_{q(n)}$ being unitary operators on $\mathcal{X} \otimes \mathcal{Y}$ induced by Toffoli, Hadamard, and $\pi/2$-phase shift gates and Λ_0 and Λ_1 being diagonal projection operators with $\Lambda_0 + \Lambda_1 = \mathbb{1}_{\mathcal{X}} \otimes \mathbb{1}_{\mathcal{Y}}$ and with diagonal entries (in $\{0, 1\}$) computable in polynomial-time. Letting r denote the number of Hadamard gates among the gates corresponding to $U_1, \ldots, U_{q(n)}$, we see that

$$\Pi_0 = \frac{1}{2^r} M_1 \cdots M_{2q(n)+1} \quad \text{and} \quad \Pi_1 = \frac{1}{2^r} N_1 \cdots N_{2q(n)+1},$$

for $M_1, \ldots, M_{2q(n)+1}$ and $N_1, \ldots, N_{2q(n)+1}$ being $2^{2m} \times 2^{2m}$ matrices whose entries have integer real and imaginary parts computable in polynomial time. By this fact it is possible to compute explicit descriptions of these matrices in NC(*poly*) by constructing an exponential-size, polynomial-depth circuit that independently computes each matrix entry of $M_1, \ldots, M_{2q(n)+1}$ and $N_1, \ldots, N_{2q(n)+1}$ in parallel, and then composing this computation with an NC algorithm for computing iterated matrix products. One obtains an NC(*poly*) algorithm for computing Π_0 and Π_1, and by performing this computation for both of the verifier's measurements one obtains an NC(*poly*) algorithm for computing Π_0^0, Π_1^0, Π_0^1, and Π_1^1.

The second stage is more difficult, and the algorithm for accomplishing it is representative of the main technical contribution of this work. The section following this one is devoted to a discussion of this algorithm. Assuming the results reported therein we have that the composition of the two steps of our computation is an NC(*poly*) computation, and therefore $A \in \text{NC}(poly) = \text{PSPACE}$.

4. A PARALLEL SEMIDEFINITE PROGRAMMING ALGORITHM

We now present a parallel algorithm that operates as follows. It takes as input four projection operators $\Pi_0^0,\ \Pi_1^0,\ \Pi_0^1,\ \Pi_1^1 \in \text{Pos}(\mathcal{X} \otimes \mathcal{Y})$ satisfying

$$\Pi_0^0 + \Pi_1^0 = \mathbb{1}_{\mathcal{X}} \otimes \mathbb{1}_{\mathcal{Y}} = \Pi_0^1 + \Pi_1^1,$$

where $\mathcal{X}$ and $\mathcal{Y}$ are complex vector spaces, both having dimension $M = 2^m$. Under the promise that the semidefinite program SDP(P_0, P_1) defined in Section 2.5, for $P_0 = \Pi_1^0 + \alpha\Pi_0^0$, $P_1 = \Pi_1^1 + \alpha\Pi_0^1$, and $\alpha = 4$, has an optimal value that is either at least 1 or less than 5/8, the algorithm determines which is the case (accepting if the optimal value is at least 1 and rejecting when it is less than 5/8).

The algorithm refers to constant values

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

which, for the sake of clarity, are referred to by these names rather than their numerical values. The algorithm is as follows.

(1) Compute

$$P_a = \Pi_1^a + \alpha \Pi_0^a, \quad X_a^{(0)} = \mathbb{1}_{\mathcal{X}} \otimes \mathbb{1}_{\mathcal{Y}}, \quad \text{and} \quad \rho_a^{(0)} = \frac{X_a^{(0)}}{\mathrm{Tr}\left(X_0^{(0)} + X_1^{(0)}\right)}$$

for $a \in \{0, 1\}$, and compute

$$Y^{(0)} = \mathbb{1}_{\mathcal{X}}, \qquad \sigma^{(0)} = \frac{Y^{(0)}}{\mathrm{Tr}\left(Y^{(0)}\right)}, \qquad \text{and} \qquad T = \left\lceil \frac{4 \ln(M)}{\varepsilon^2 \delta} \right\rceil.$$

(2) Repeat the following steps for each t from 0 to $T - 1$:
   (a) Compute

$$Z_a^{(t)} = \frac{\gamma}{2} \sigma^{(t)} - \mathrm{Tr}_{\mathcal{Y}} \left(P_a \rho_a^{(t)} P_a\right)$$

   and compute the projection $\Delta_a^{(t)}$ onto the negative eigenspace of $Z_a^{(t)}$, for $a \in \{0, 1\}$.
   (b) Compute

$$\beta^{(t)} = \left\langle \Delta_0^{(t)} \otimes \mathbb{1}_{\mathcal{Y}}, P_0 \rho_0^{(t)} P_0 \right\rangle + \left\langle \Delta_1^{(t)} \otimes \mathbb{1}_{\mathcal{Y}}, P_1 \rho_1^{(t)} P_1 \right\rangle.$$

   If $\beta^{(t)} \leq \varepsilon$, then output 1 (i.e., *accept*) and stop.
   (c) Compute

$$X_a^{(t+1)} = \exp\left(-\sum_{j=0}^{t} \frac{\varepsilon \delta}{\beta^{(j)}} P_a \left(\Delta_a^{(j)} \otimes \mathbb{1}_{\mathcal{Y}}\right) P_a\right)$$

$$\rho_a^{(t+1)} = \frac{X_a^{(t+1)}}{\mathrm{Tr}\left(X_0^{(t+1)} + X_1^{(t+1)}\right)}$$

   for $a \in \{0, 1\}$ and

$$Y^{(t+1)} = \exp\left(\sum_{j=0}^{t} \frac{\varepsilon^2}{2\beta^{(j)}} \left(\Delta_0^{(j)} + \Delta_1^{(j)}\right)\right)$$

$$\sigma^{(t+1)} = \frac{Y^{(t+1)}}{\mathrm{Tr}\left(Y^{(t+1)}\right)}.$$

(3) Output 0 (i.e., *reject*).

4.1. Intuition behind the Algorithm

Before presenting the analysis of the algorithm, we briefly discuss the intuition behind the algorithm itself.

First, it should be noted that the algorithm does not really attempt to approximate the optimal value of $\mathrm{SDP}(P_0, P_1)$, but merely attempts to determine whether its optimal value is greater or less than $1/\gamma = 3/4$. One could repeat the algorithm for different values of γ and ε for an increasingly accurate approximation of the optimum, but this is not necessary for solving the decision problem under consideration.

The variables $\rho_0^{(t)}$, $\rho_1^{(t)}$, and $\sigma^{(t)}$ generated in successive iterations within the algorithm will be used to specify possible candidates for Q_0, Q_1, and ξ in SDP(P_0, P_1). More specifically, if it is the case that the substitutions $Q_0 = 2\rho_0^{(t)}$, $Q_1 = 2\rho_1^{(t)}$, and $\xi = \sigma^{(t)}$ yield a setting of (Q_0, Q_1, ξ) that is *close* to satisfying the primal constraints of SDP(P_0, P_1), then the algorithm accepts.

The specific notion of closeness used by the algorithm is represented by the number $\beta^{(t)}$, whose specification is technical but well-suited to the analysis of the algorithm. At an informal and intuitive level, one may observe that large constraint violations are required for the operators $Z_0^{(t)}$ and $Z_1^{(t)}$ to have nontrivial negative eigenspaces, and $\beta^{(t)}$ may be seen as a sort of measure of these spaces relative to the current choices of $\rho_0^{(t)}$ and $\rho_1^{(t)}$. It is not difficult to show, as is done in the formal analysis that follows in the next section, that if $\rho_0^{(t)}$, $\rho_1^{(t)}$, and $\sigma^{(t)}$ provide a setting of (Q_0, Q_1, ξ) that is close to being feasible, then these operators can be massaged to yield a truly feasible solution that achieves a large objective value for the primal problem.

If it so happens that a given choice of $\rho_0^{(t)}$, $\rho_1^{(t)}$, and $\sigma^{(t)}$ fails to yield a setting for (Q_0, Q_1, ξ) that is close to meeting the primal constraints of SDP(P_0, P_1), then the algorithm has effectively generated a witness of this failure: $\Delta_0^{(t)}$ and $\Delta_1^{(t)}$. The operators $\Delta_0^{(t)}/\beta^{(t)}$ and $\Delta_1^{(t)}/\beta^{(t)}$ are then implicitly used to construct candidate settings for the dual variables R_0 and R_1, respectively. The quality of the dual variables R_0 and R_1 depends on the number of iterations of the algorithm that are performed: if a logarithmic number T of iterations pass in which the algorithm fails to find a setting of (Q_0, Q_1, ξ) that is close to being primal feasible, a dual-feasible setting for R_0 and R_1 achieving a small dual objective value can be exhibited. It is critical for the parallelizability of the algorithm that a logarithmic setting for T (which corresponds to a polynomial in the input size to the original problem $A \in$ QIP) suffices for this demonstration.

The magic of the matrix multiplicative weights update method takes place in step (2)(c), where $\rho_0^{(t+1)}$, $\rho_1^{(t+1)}$, and $\sigma^{(t+1)}$ are regenerated from the previously computed values of $\Delta_0^{(j)}/\beta^{(j)}$ and $\Delta_1^{(j)}/\beta^{(j)}$, for $j = 0, \ldots, t$. Lemma 2 from Section 2 provides the key technical tool that allows for a formal demonstration that the algorithm operates correctly by means of these regenerations.

4.2. Analysis (Ignoring Precision)

We now show that the algorithm described previously answers correctly, meaning (1) if the optimal value of the semidefinite program SDP(P_0, P_1) is at least 1, then the algorithm accepts, and (2) if the optimal value is at most 5/8, then the algorithm rejects. These implications are established by considering two cases: one is that the algorithm accepts and the other is that the algorithm rejects. The behavior of the algorithm in these cases will allow us to construct feasible solutions to SDP(P_0, P_1) that establish the required implications.

In this section we consider only 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 section following this one.

Case 1. (*The Algorithm Accepts*). If the algorithm accepts, it must do so during an iteration of step (2). For whichever iteration t it is, we will write ρ_0, ρ_1, σ, Δ_0, Δ_1, and β rather than $\rho_0^{(t)}$, $\rho_1^{(t)}$, $\sigma^{(t)}$, $\Delta_0^{(t)}$, $\Delta_1^{(t)}$, and $\beta^{(t)}$ to simplify our notation. Define

$$Q_a = \frac{2\rho_a}{\gamma + 2\beta} \in \mathrm{Pos}(\mathcal{X} \otimes \mathcal{Y})$$

for $a \in \{0, 1\}$, and define

$$\xi = \frac{\gamma\sigma + 2\Delta_0 \operatorname{Tr}_{\mathcal{Y}}(P_0\rho_0P_0)\Delta_0 + 2\Delta_1 \operatorname{Tr}_{\mathcal{Y}}(P_1\rho_1P_1)\Delta_1}{\gamma + 2\beta} \in \mathrm{D}(\mathcal{X}).$$

The fact that ξ is a density operator is easily verified. We will prove that (Q_0, Q_1, ξ) is a primal feasible point of SDP(P_0, P_1) that achieves an objective value larger than 5/8.

For each $a \in \{0, 1\}$, it holds that

$$-\Delta_a \operatorname{Tr}_{\mathcal{Y}}(P_a\rho_aP_a)\Delta_a \leq \Delta_a \left(\frac{\gamma}{2}\sigma - \operatorname{Tr}_{\mathcal{Y}}(P_a\rho_aP_a)\right)\Delta_a \leq \frac{\gamma}{2}\sigma - \operatorname{Tr}_{\mathcal{Y}}(P_a\rho_aP_a),$$

with the first inequality following from the fact that $\frac{\gamma}{2}\Delta_a\sigma\Delta_a$ is positive semidefinite, and with the second inequality holding by the definition of Δ_a. We conclude that

$$\operatorname{Tr}_{\mathcal{Y}}(P_a\rho_aP_a) \leq \frac{\gamma}{2}\sigma + \Delta_a \operatorname{Tr}_{\mathcal{Y}}(P_a\rho_aP_a)\Delta_a, \tag{10}$$

and thus

$$\operatorname{Tr}_{\mathcal{Y}}(P_aQ_aP_a) \leq \frac{\gamma\sigma + 2\Delta_a \operatorname{Tr}_{\mathcal{Y}}(P_a\rho_aP_a)\Delta_a}{\gamma + 2\beta} \leq \xi.$$

It is therefore established that (Q_0, Q_1, ξ) is a feasible solution to the primal problem of SDP(P_0, P_1).

Given that the algorithm has accepted it holds that $\beta \leq \varepsilon$, and therefore the primal objective value obtained is

$$\operatorname{Tr}\left(\frac{1}{2}Q_0 + \frac{1}{2}Q_1\right) = \frac{1}{\gamma + 2\beta} \geq \frac{1}{\gamma + 2\epsilon} > \frac{5}{8}.$$

Consequently, if the optimal value of SDP(P_0, P_1) is at most 5/8, then the algorithm cannot accept, so it must reject.

Case 2. (*The Algorithm Rejects*). Now assume that the algorithm rejects, and define

$$R_a = \frac{1 + 4\varepsilon}{T} \sum_{t=0}^{T-1} \frac{1}{\beta^{(t)}} \Delta_a^{(t)} \tag{11}$$

for $a \in \{0, 1\}$. We will prove that the pair (R_0, R_1) forms a feasible solution to the dual problem of SDP(P_0, P_1) that achieves an objective value smaller than 7/8.

Toward this goal, define a $(2M^2) \times (2M^2)$ matrix

$$M^{(t)} = \frac{\delta}{\beta^{(t)}} \begin{pmatrix} P_0 \left(\Delta_0^{(t)} \otimes \mathbb{1}_{\mathcal{Y}}\right) P_0 & 0 \\ 0 & P_1 \left(\Delta_1^{(t)} \otimes \mathbb{1}_{\mathcal{Y}}\right) P_1 \end{pmatrix} \tag{12}$$

for each $t \in \{0, \ldots, T-1\}$. Each $M^{(t)}$ is positive semidefinite, and under the assumption that the algorithm rejects it holds that $\beta^{(t)} \geq \varepsilon$ and therefore, by the submultiplicativity of the spectral norm, $M^{(t)} \leq \mathbb{1}$. Writing

$$X^{(t)} = \begin{pmatrix} X_0^{(t)} & 0 \\ 0 & X_1^{(t)} \end{pmatrix} \quad \text{and} \quad \rho^{(t)} = \begin{pmatrix} \rho_0^{(t)} & 0 \\ 0 & \rho_1^{(t)} \end{pmatrix}$$

for $t \in \{0, \ldots, T-1\}$, one sees that item (1) of Lemma 2 applies (for $N = 2M^2$), and implies that

$$\lambda_{\min}\left(\sum_{t=0}^{T-1} M^{(t)}\right) \geq \exp(-\varepsilon)\sum_{t=0}^{T-1}\left\langle \rho^{(t)}, M^{(t)}\right\rangle - \frac{\ln(2M^2)}{\varepsilon}. \tag{13}$$

For each $t \in \{0, \ldots, T-1\}$, it holds that $\left\langle \rho^{(t)}, M^{(t)}\right\rangle = \delta$, and therefore

$$\lambda_{\min}\left(\sum_{t=0}^{T-1} M^{(t)}\right) \geq \exp(-\varepsilon)\delta T - \frac{\ln(2M^2)}{\varepsilon},$$

so that

$$\lambda_{\min}\begin{pmatrix} P_0(R_0 \otimes \mathbb{1}_{\mathcal{Y}})P_0 & 0 \\ 0 & P_1(R_1 \otimes \mathbb{1}_{\mathcal{Y}})P_1 \end{pmatrix} \geq (1+4\varepsilon)\left(\exp(-\varepsilon) - \frac{\ln(2M^2)}{\varepsilon\delta T}\right) \geq 1.$$

Equivalently, $P_a(R_a \otimes \mathbb{1}_{\mathcal{Y}})P_a \geq \mathbb{1}_{\mathcal{X}} \otimes \mathbb{1}_{\mathcal{Y}}$ for $a \in \{0,1\}$, implying that (R_0, R_1) forms a dual feasible solution to SDP(P_0, P_1) as claimed.

It remains to prove an upper bound on the dual objective value achieved by (R_0, R_1). To do this, we will apply item (2) of Lemma 2, this time taking $N = M$ and

$$M^{(t)} = \frac{\varepsilon}{2\beta^{(t)}}\left(\Delta_0^{(t)} + \Delta_1^{(t)}\right), \tag{14}$$

which is consistent with the algorithm's computation of $Y^{(t)}$ and $\sigma^{(t)}$. The assumption that the algorithm rejects implies that $\beta^{(t)} > \varepsilon$, and therefore $M^{(t)} \leq \mathbb{1}_{\mathcal{X}}$ for each $t \in \{0, \ldots, T-1\}$. Lemma 2 therefore implies that

$$\lambda_{\max}\left(\sum_{t=0}^{T-1} M^{(t)}\right) \leq \exp(\varepsilon)\sum_{t=0}^{T-1}\left\langle \sigma^{(t)}, M^{(t)}\right\rangle + \frac{\ln(M)}{\varepsilon}.$$

To upper-bound the right-hand-side of this expression, we note that

$$\Delta_a^{(t)}\left(\frac{\gamma}{2}\sigma^{(t)} - \mathrm{Tr}_{\mathcal{Y}}\left(P_a\rho_a^{(t)}P_a\right)\right)\Delta_a^{(t)} \leq 0$$

and therefore

$$\frac{\gamma}{2}\left\langle \Delta_a^{(t)}, \sigma^{(t)}\right\rangle \leq \left\langle \Delta_a^{(t)}, \mathrm{Tr}_{\mathcal{Y}}\left(P_a\rho_a^{(t)}P_a\right)\right\rangle = \left\langle \Delta_a^{(t)} \otimes \mathbb{1}_{\mathcal{Y}}, P_a\rho_a^{(t)}P_a\right\rangle,$$

for $a \in \{0,1\}$. It follows that $\left\langle \sigma^{(t)}, M^{(t)}\right\rangle \leq \varepsilon/\gamma$ for each $t \in \{0, \ldots, T-1\}$, so that

$$\lambda_{\max}\left(\sum_{t=0}^{T-1} M^{(t)}\right) \leq \frac{\varepsilon\exp(\varepsilon)T}{\gamma} + \frac{\ln(M)}{\varepsilon},$$

and therefore

$$\left\|\frac{1}{2}R_0 + \frac{1}{2}R_1\right\| = \frac{1+4\varepsilon}{\varepsilon T}\lambda_{\max}\left(\sum_{t=0}^{T-1} M^{(t)}\right) \leq (1+4\varepsilon)\left(\frac{\exp(\varepsilon)}{\gamma} + \frac{\ln(M)}{\varepsilon^2 T}\right) < \frac{7}{8}.$$

If the optimal value of SDP(P_0, P_1) is at least 1, it therefore cannot be that the algorithm rejects, so it must accept. Thus, the algorithm operates as required.

4.3. Comments on Precision

The algorithm computes operators $\Delta_a^{(t)}$, $\rho_a^{(t)}$, and $\sigma^{(t)}$, as well as scalar values $\beta^{(t)}$, for $a \in \{0,1\}$ and some range of values for t. The algorithm also computes $Z_a^{(t)}$, $X_a^{(t)}$, and $Y^{(t)}$ for $a \in \{0,1\}$ and varying values of t, but these operators are only used locally within the algorithm in what may be viewed as intermediate calculations leading to $\Delta_a^{(t)}$, $\rho_a^{(t)}$, $\sigma^{(t)}$, and $\beta^{(t)}$. With this fact in mind, for the sake of presenting the analysis that follows in simple terms, we will take each operator $Z_a^{(t)}$, $X_a^{(t)}$, and $Y^{(t)}$ to be an idealized operator that would result from an exact computation. In other words, we define these operators as $X_a^{(0)} = \mathbb{1}_{\mathcal{X}} \otimes \mathbb{1}_{\mathcal{Y}}$, $Y^{(0)} = \mathbb{1}_{\mathcal{X}}$,

$$X_a^{(t+1)} = \exp\left(-\sum_{j=0}^{t} \frac{\varepsilon\delta}{\beta^{(j)}} P_a \left(\Delta_a^{(j)} \otimes \mathbb{1}_{\mathcal{Y}}\right) P_a\right),$$

$$Y^{(t+1)} = \exp\left(\sum_{j=0}^{t} \frac{\varepsilon^2}{2\beta^{(j)}} \left(\Delta_0^{(j)} + \Delta_1^{(j)}\right)\right),$$

and

$$Z_a^{(t)} = \frac{\gamma}{2}\sigma^{(t)} - \mathrm{Tr}_{\mathcal{Y}}\left(P_a \rho_a^{(t)} P_a\right),$$

for $a \in \{0,1\}$ and $t \in \{0,\ldots,T-1\}$, where $\Delta_a^{(t)}$, $\rho_a^{(t)}$, $\sigma^{(t)}$, and $\beta^{(t)}$ refer to the finite-precision operators and scalars that are actually stored by an implementation of the algorithm. Let us also define $\Lambda_a^{(t)}$ to be the true projection operator onto the negative eigenspace of $Z_a^{(t)}$ for each $a \in \{0,1\}$ and $t \in \{0,\ldots,T-1\}$.

Now, the matrix exponentials can only be computed approximately, and as a result one can only guarantee that

$$\rho_a^{(t)} \approx \frac{X_a^{(t)}}{\mathrm{Tr}\left(X_0^{(t)} + X_1^{(t)}\right)} \quad \text{and} \quad \sigma^{(t)} \approx \frac{Y^{(t)}}{\mathrm{Tr}\left(Y^{(t)}\right)}.$$

Similarly, the negative eigenspace computations can only be performed approximately, so that $\Delta_a^{(t)} \approx \Lambda_a^{(t)}$. The precise assumptions we will make on the accuracy of these approximations, to be justified in the section following this one, are as follows:

(a) For each $a \in \{0,1\}$ and $t \in \{0,\ldots,T-1\}$, $\Delta_a^{(t)}$ is a positive semidefinite operator that satisfies

$$\left\| \sqrt{\Delta_a^{(t)}} - \Lambda_a^{(t)} \right\| < \frac{\varepsilon^2}{2\alpha^2 M}. \tag{15}$$

We will also assume (for the sake of convenience more than necessity) that each $\Delta_a^{(t)}$ satisfies $\Delta_a^{(t)} \leq \mathbb{1}_{\mathcal{X}}$. Under these assumption, one may conclude that

$$\sqrt{\Delta_a^{(t)}} Z_a^{(t)} \sqrt{\Delta_a^{(t)}} \leq \Lambda_a^{(t)} Z_a^{(t)} \Lambda_a^{(t)} + \frac{\varepsilon^2}{M} \mathbb{1}_{\mathcal{X}}. \tag{16}$$

(b) For each $a \in \{0,1\}$ and $t \in \{0,\ldots,T-1\}$, it holds that $\rho_a^{(t)}$ and $\sigma^{(t)}$ are positive semidefinite operators that satisfy

$$\left\| \rho_a^{(t)} - \frac{X_a^{(t)}}{\mathrm{Tr}\left(X_0^{(t)} + X_1^{(t)}\right)} \right\| < \frac{\varepsilon\delta}{2M^2} \tag{17}$$

and

$$\left\| \sigma^{(t)} - \frac{Y^{(t)}}{\mathrm{Tr}\left(Y^{(t)}\right)} \right\| < \frac{\varepsilon^2}{\gamma M}. \tag{18}$$

In addition, we will assume that both $\rho_0^{(t)} + \rho_1^{(t)}$ and $\sigma^{(t)}$ have trace equal to 1 (and are therefore density operators).

The scalars $\beta^{(t)}$ may be computed exactly, meaning that

$$\beta^{(t)} = \left\langle \Delta_0^{(t)} \otimes \mathbb{1}_{\mathcal{Y}}, P_0 \rho_0^{(t)} P_0 \right\rangle + \left\langle \Delta_1^{(t)} \otimes \mathbb{1}_{\mathcal{Y}}, P_1 \rho_1^{(t)} P_1 \right\rangle$$

for $t = 0,\ldots,T-1$, so it is not necessary to make an assumption on their accuracy.

As in the analysis for the exact case, one may consider two cases when the computations are inexact: one is that the algorithm accepts and the other is that the algorithm rejects. The aspects of the analysis that differ from the exact case are as follows.

Case 1. (*The Algorithm Accepts*). As in the exact case, we write ρ_0, ρ_1, σ, Δ_0, Δ_1, and β rather than $\rho_0^{(t)}$, $\rho_1^{(t)}$, $\sigma^{(t)}$, $\Delta_0^{(t)}$, $\Delta_1^{(t)}$, and $\beta^{(t)}$ for t corresponding to the iteration in which acceptance occurs. This time, we define

$$Q_a = \frac{2\rho_a}{\gamma + 2\beta + \varepsilon}$$

for $a \in \{0,1\}$, and

$$\xi = \frac{\gamma\sigma + 2\sqrt{\Delta_0}\,\mathrm{Tr}_{\mathcal{Y}}(P_0\rho_0P_0)\sqrt{\Delta_0} + 2\sqrt{\Delta_1}\,\mathrm{Tr}_{\mathcal{Y}}(P_1\rho_1P_1)\sqrt{\Delta_1} + \frac{\varepsilon}{M}\mathbb{1}_{\mathcal{X}}}{\gamma + 2\beta + \varepsilon}$$

(which, once again, is a density operator).

In place of the inequality (10), one may obtain the inequality

$$\mathrm{Tr}_{\mathcal{Y}}(P_a\rho_aP_a) \leq \frac{\gamma}{2}\sigma + \sqrt{\Delta_a}\,\mathrm{Tr}_{\mathcal{Y}}(P_a\rho_aP_a)\sqrt{\Delta_a} + \frac{\varepsilon^2}{M}\mathbb{1}_{\mathcal{X}} \tag{19}$$

by means of (16). It follows that

$$\mathrm{Tr}_{\mathcal{Y}}(P_aQ_aP_a) \leq \frac{\gamma\sigma + 2\sqrt{\Delta_a}\,\mathrm{Tr}_{\mathcal{Y}}(P_a\rho_aP_a)\sqrt{\Delta_a} + \frac{2\varepsilon^2}{M}\mathbb{1}_{\mathcal{X}}}{\gamma + 2\beta + \varepsilon} \leq \xi,$$

so (Q_0, Q_1, ξ) is a feasible solution to the primal problem of SDP(P_0, P_1). Given that the algorithm has accepted it holds that $\beta \leq \varepsilon$, and therefore the primal objective value obtained is

$$\mathrm{Tr}\left(\frac{1}{2}Q_0 + \frac{1}{2}Q_1\right) = \frac{1}{\gamma + 2\beta + \varepsilon} \geq \frac{1}{\gamma + 3\epsilon} > \frac{5}{8}.$$

Case 2. (*The Algorithm Rejects*). In the case that the algorithm rejects, we once again consider the pair (R_0, R_1) defined precisely as in the error-free case (as given by (11)). Lemma 2 is applied to the operators $M^{(0)}, \ldots, M^{(T-1)}$ as defined in (12), yielding the bound

$$\lambda_{\min}\left(\sum_{t=0}^{T-1} M^{(t)}\right) \geq \exp(-\varepsilon)\sum_{t=0}^{T-1}\left\langle \frac{X^{(t)}}{\mathrm{Tr}\left(X^{(t)}\right)}, M^{(t)}\right\rangle - \frac{\ln(2M^2)}{\varepsilon} \tag{20}$$

in place of (13). By means of (17) we obtain

$$\left\langle \frac{X^{(t)}}{\mathrm{Tr}\left(X^{(t)}\right)}, M^{(t)}\right\rangle \geq (1-\varepsilon)\delta,$$

which yields

$$\lambda_{\min}\begin{pmatrix} P_0(R_0 \otimes \mathbb{1}_{\mathcal{Y}})P_0 & 0 \\ 0 & P_1(R_1 \otimes \mathbb{1}_{\mathcal{Y}})P_1 \end{pmatrix} \geq (1+4\varepsilon)\left(\exp(-\varepsilon)(1-\varepsilon) - \frac{\ln(2M^2)}{\varepsilon\delta T}\right) \geq 1,$$

by similar reasoning to the error-free analysis. Thus, dual feasibility holds for (R_0, R_1).

Now, when bounding the dual objective value obtained by (R_0, R_1), we once again apply Lemma 2 to the operators $M^{(0)}, \ldots, M^{(T-1)}$ as defined in (14), obtaining

$$\lambda_{\max}\left(\sum_{t=0}^{T-1} M^{(t)}\right) \leq \exp(\varepsilon)\sum_{t=0}^{T-1}\left\langle \frac{Y^{(t)}}{\mathrm{Tr}\left(Y^{(t)}\right)}, M^{(t)}\right\rangle + \frac{\ln(M)}{\varepsilon}.$$

By means of first (18) and then (16), we have

$$\left\langle \frac{Y^{(t)}}{\mathrm{Tr}\left(Y^{(t)}\right)}, M^{(t)}\right\rangle \leq \left\langle \sigma^{(t)}, M^{(t)}\right\rangle + \frac{\varepsilon^2}{\gamma} \leq \frac{(1+2\varepsilon)\varepsilon}{\gamma},$$

which implies

$$\left\| \frac{1}{2}R_0 + \frac{1}{2}R_1 \right\| \leq (1+4\varepsilon)\left(\frac{\exp(\varepsilon)(1+2\varepsilon)}{\gamma} + \frac{\ln(M)}{\varepsilon^2 T}\right) < \frac{7}{8}.$$

4.4. An NC Implementation of the Algorithm

In this section, we explain how the previous algorithm can be implemented by an NC computation.

The matrices stored and processed by our algorithm will have entries with rational real and imaginary parts, and we assume that all rational numbers are represented as pairs of integers in binary notation. 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 [1993] for more details.

As was mentioned in the previous section, the negative eigenspace computations in step (2)(a) and the matrix exponential computations in step (2)(c) of the algorithm will not be performed exactly in our implementation. Based on the previous discussion

of precision, however, it will suffice that the following computational problems can be solved by NC computations.

Matrix exponentials

Input: An $n \times n$ matrix A, a positive rational number η, and an integer k expressed in unary notation (i.e., 1^k).

Promise: $\|A\| \leq k$.

Output: An $n \times n$ matrix X such that $\left\|\exp(A) - X\right\| < \eta$.

Negative 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 negative 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

$$\exp(A) = \sum_{m=0}^{\infty} \frac{A^m}{m!}$$

to a number of terms linear in $k + \ln(1/\eta)$. (From a numerical point of view this is not necessarily a good way to compute matrix exponentials [Moler and Van Loan 2003], but it is arguably the simplest way to prove that the stated problem is in NC.) In the case that A is Hermitian, it holds that $\exp(A)$ 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 $\exp(A)$.

That negative 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 by Ben-Or et al. [1986], and an alternate proof was given by Bini and Pan [1998]. While the real-root case is sufficient for our needs, we note that the general case, which allows for complex roots, was solved by Neff [1994]. It is important to note that distinct nonzero roots of integer polynomials can neither be too close to one another nor to zero [Mahler 1961; Bugeaud 2004], 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 negative roots (in the real-root case).

Given that the characteristic polynomial of a matrix can be computed in NC [Csanky 1976], 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 simple way to avoid the special case where $\kappa = \lambda$ is to perturb κ slightly.) By

summing the approximate projections onto the eigenspaces corresponding to distinct negative eigenvalues of H, we obtain a close approximation Δ to Λ as claimed. (The fact that our analysis places a requirement on $\|\sqrt{\Delta} - \Lambda\|$ rather than $\|\Delta - \Lambda\|$ is easily handled by computing an approximation Δ to Λ, then substituting Δ^2 for Δ in the algorithm.)

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 $\Delta_a^{(t)}$, $\rho_a^{(t)}$ and $\sigma^{(t)}$ is to be stored by the algorithm as an integer divided by 2^K, for $K = O(M)$ (or even $K = O(\log M)$ 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 M, 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. OPEN PROBLEMS

We will conclude this article 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 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. Other algorithms appearing in previous works, including both sequential algorithms [Arora and Kale 2007; Arora et al. 2005] and parallel algorithms [Jain and Watrous 2009; Jain et al. 2009], for efficiently solving semidefinite programs using the matrix multiplicative weights update method have also been limited to specific subclasses of semidefinite programs. To what extent these methods can be applied to more general classes of semidefinite programs remains open. 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 [Dobkin et al. 1979; Megiddo 1992; Serna 1991]. (Recent progress on this question has been reported by Jain and Yao [2011].)

A second open problem relating to this article 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 [Gutoski and Watrous 2007] and the classical analogue of this class is known to coincide with PSPACE [Feige and Kilian 1997]. (A positive answer to this question has recently been reported by Gutoski and Wu [2010].)

ACKNOWLEDGMENTS

We thank Xiaodi Wu for helpful discussions, and we thank several anonymous referees for numerous suggestions for improving on earlier versions of this article.

REFERENCES

ARORA, S. AND BARAK, B. 2009. *Computational Complexity: A Modern Approach*. Cambridge University Press.

ARORA, S., HAZAN, E., AND KALE, S. 2005. 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*. IEEE Computer Society Press, Los Alamitos, CA, 339–348.

ARORA, S. AND KALE, S. 2007. A combinatorial, primal-dual approach to semidefinite programs. In *Proceedings of the 39th Annual ACM Symposium on Theory of Computing*. ACM, New York, 227–236.

BABAI, L. 1985. Trading group theory for randomness. In *Proceedings of the 17th Annual ACM Symposium on Theory of Computing*. ACM, New York, 421–429.

BABAI, L. AND MORAN, S. 1988. Arthur-Merlin games: A randomized proof system, and a hierarchy of complexity classes. *J. Comput. Syst. Sci. 36,* 2, 254–276.

BABAI, L., FORTNOW, L., AND LUND, C. 1991. Non-deterministic exponential time has two-prover interactive protocols. *Computat. Complex. 1,* 1, 3–40.

BEN-OR, M., FEIG, E., KOZEN, D., AND TIWARI, P. 1986. 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*. ACM, New York, 340–349.

BEN-OR, M., GOLDWASSER, S., KILIAN, J., AND WIGDERSON, A. 1988. Multi-prover interactive proofs: How to remove intractability assumptions. In *Proceedings of the 20th Annual ACM Symposium on Theory of Computing*. ACM, New York, 113–131.

BENNETT, C. AND BRASSARD, G. 1984. Quantum cryptography: Public key distribution and coin tossing. In *Proceedings of the IEEE International Conference on Computers, Systems, and Signal Processing*. IEEE Computer Society Press, Los Alamitos, CA, 175–179.

BHATIA, R. 1997. *Matrix Analysis*. Springer, New York.

BINI, D. AND PAN, V. 1998. Computing matrix eigenvalues and polynomial zeros where the output is real. *SIAM J. Comput. 27,* 4, 1099–1115.

BORODIN, A. 1977. On relating time and space to size and depth. *SIAM J. Comput. 6*, 733–744.

BORODIN, A., COOK, S., AND PIPPENGER, N. 1983. Parallel computation for well-endowed rings and space-bounded probabilistic machines. *Inf. Cont. 58*, 113–136.

BUGEAUD, Y. 2004. *Approximation by Algebraic Numbers*. Cambridge Tracts in Mathematics, vol. 160. Cambridge University Press, Cambridge, UK.

BUHRMAN, H., CLEVE, R., AND WIGDERSON, A. 1998. Quantum vs. classical communication and computation. In *Proceedings of the 30th Annual ACM Symposium on Theory of Computing*. ACM, New York, 63–68.

CHURCH, A. 1936. An unsolvable problem of elementary number theory. *Amer. J. Math. 58,* 2, 345–363.

COOK, S. 1971. The complexity of theorem proving procedures. In *Proceedings of the 3rd Annual ACM Symposium on Theory of Computing*. ACM, New York, 151–158.

CSANKY, L. 1976. Fast parallel matrix inversion algorithms. *SIAM J. Comput. 5,* 4, 618–623.

DEUTSCH, D. 1985. Quantum theory, the Church–Turing principle and the universal quantum computer. *Proc. Roy. Soc. London A400*, 97–117.

DOBKIN, D., LIPTON, R., AND REISS, S. 1979. Linear programming is log-space hard for P. *Inf. Proc. Lett. 8,* 2, 96–97.

EVEN, S., SELMAN, A., AND YACOBI, Y. 1984. The complexity of promise problems with applications to public-key cryptography. *Inf. Cont. 61,* 2, 159–173.

FEIGE, U. AND KILIAN, J. 1997. Making games short. In *Proceedings of the 29th Annual ACM Symposium on Theory of Computing*. ACM, New York, 506–516.

FELDMAN, P. 1986. The optimum prover lies in PSPACE. Manuscript.

FEYNMAN, R. 1982. Simulating physics with computers. *Int. J. Theoret. Physics 21,* 6/7, 467–488.

FURER, M., GOLDREICH, O., MANSOUR, Y., SIPSER, M., AND ZACHOS, S. 1989. On completeness and soundness in interactive proof systems. In *Randomness and Computation*, S. Micali Ed. Advances in Computing Research, vol. 5. JAI Press, 429–442.

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

GOLDREICH, O., MICALI, S., AND WIGDERSON, A. 1991. Proofs that yield nothing but their validity or all languages in NP have zero-knowledge proof systems. *J. ACM 38,* 1, 691–729.

GOLDWASSER, S. AND SIPSER, M. 1989. Private coins versus public coins in interactive proof systems. In *Randomness and Computation*, S. Micali Ed. Advances in Computing Research, vol. 5. JAI Press, 73–90.

GOLDWASSER, S., MICALI, S., AND RACKOFF, C. 1985. The knowledge complexity of interactive proof systems. In *Proceedings of the 17th Annual ACM Symposium on Theory of Computing*. ACM, New York, 291–304.

GOLDWASSER, S., MICALI, S., AND RACKOFF, C. 1989. The knowledge complexity of interactive proof systems. *SIAM J. Comput. 18,* 1, 186–208.

GUTOSKI, G. AND WATROUS, J. 2007. Toward a general theory of quantum games. In *Proceedings of the 39th Annual ACM Symposium on Theory of Computing*. ACM, New York, 565–574.

GUTOSKI, G. AND WU, X. 2010. Short quantum games characterize PSPACE. arXiv.org e-Print 1011.2787.

HALLGREN, S., KOLLA, A., SEN, P., AND ZHANG, S. 2008. Making classical honest verifier zero knowledge protocols secure against quantum attacks. In *Proceedings of the 35th International Colloquium on Automata, Languages and Programming*. Lecture Notes in Computer Science, vol. 5126. Springer, 592–603.

HORODECKI, R., HORODECKI, P., HORODECKI, M., AND HORODECKI, K. 2009. Quantum entanglement. *Rev. Modern Phys. 81,* 2, 865–942.

JAIN, R. AND WATROUS, J. 2009. Parallel approximation of non-interactive zero-sum quantum games. In *Proceedings of the 24th IEEE Conference on Computational Complexity*. IEEE Computer Society Press, Los Alamitos, CA, 243–253.

JAIN, R. AND YAO, P. 2011. A parallel approximation algorithm for positive semidefinite programming. In *Proceedings of the 52nd Annual IEEE Symposium on Foundations of Computer Science*. IEEE Computer Society Press, Los Alamitos, CA. To appear.

JAIN, R., UPADHYAY, S., AND WATROUS, J. 2009. Two-message quantum interactive proofs are in PSPACE. In *Proceedings of the 50th Annual IEEE Symposium on Foundations of Computer Science*. IEEE Computer Society Press, Los Alamitos, CA, 534–543.

KALE, S. 2007. Efficient algorithms using the multiplicative weights update method. Ph.D. thesis, Princeton University.

KARP, R. 1972. Reducibility among combinatorial problems. In *Complexity of Computer Computations*, R. Miller and J. Thatcher Eds. Plenum Press, New York, 85–103.

KEMPE, J., KOBAYASHI, H., MATSUMOTO, K., AND VIDICK, T. 2009. Using entanglement in quantum multi-prover interactive proofs. *Computat. Complex. 18,* 2, 273–307.

KITAEV, A. AND WATROUS, J. 2000. Parallelization, amplification, and exponential time simulation of quantum interactive proof system. In *Proceedings of the 32nd Annual ACM Symposium on Theory of Computing*. ACM, New York, 608–617.

KOBAYASHI, H. 2008. General properties of quantum zero-knowledge proofs. In *Proceedings of the 5th IACR Theory of Cryptography Conference*. Lecture Notes in Computer Science, vol. 4948. Springer, Berlin, 107–124.

KOBAYASHI, H. AND MATSUMOTO, K. 2003. Quantum multi-prover interactive proof systems with limited prior entanglement. *J. Comput. Syst. Sci. 66,* 3, 429–450.

LEVIN, L. 1973. Universal sequential search problems (English translation). *Probl. Inf. Trans. 9,* 3, 265–266.

LUND, C., FORTNOW, L., KARLOFF, H., AND NISAN, N. 1992. Algebraic methods for interactive proof systems. *J. ACM 39,* 4, 859–868.

MAHLER, K. 1961. *Lectures on Diophantine Approximations*. Cushing Malloy, Ann Arbor, MI.

MARRIOTT, C. AND WATROUS, J. 2005. Quantum Arthur-Merlin games. *Computat. Complex. 14,* 2, 122–152.

MEGIDDO, N. 1992. A note on approximate linear programming. *Inf. Proc. Lett. 42,* 1, 53.

MOLER, C. AND VAN LOAN, C. 2003. Nineteen dubious ways to compute the exponential of a matrix, twenty-five years later. *SIAM Review 45,* 1, 3–49.

NEFF, C. A. 1994. Specified precision polynomial root isolation is in NC. *J. Comput. Syst. Sci. 48,* 3, 429–463.

NIELSEN, M. AND CHUANG, I. 2000. *Quantum Computation and Quantum Information*. Cambridge University Press, Cambridge, UK.

RAZ, R. 1999. Exponential separation of quantum and classical communication complexity. In *Proceedings of the 31st Annual ACM Symposium on Theory of Computing*. ACM, New York, 358–376.

SERNA, M. 1991. Approximating linear programming is log-space complete for P. *Inf. Proc. Lett. 37,* 4, 233–236.

SHAMIR, A. 1992. IP = PSPACE. *J. ACM 39,* 4, 869–877.

SHOR, P. 1997. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM J. Comput. 26,* 5, 1484–1509.

SIPSER, M. 1997. *Introduction to the Theory of Computation*. PWS Publishing Company, Boston, MA.

TERHAL, B. 2004. Is entanglement monogamous? *IBM J. Res. Develop. 48,* 1, 71–78.

TURING, A. 1937. On computable numbers, with an application to the Entscheidungsproblem. *Proc. London Math. Soc.* (second series) *42,* 1, 230–265.

VON ZUR GATHEN, J. 1993. Parallel linear algebra. In *Synthesis of Parallel Algorithms*. J. Reif Ed. Morgan Kaufmann Publishers, Inc., San Francisco, CA, Chap. 13, 573–618.

WARMUTH, M. AND KUZMIN, D. 2006. Online variance minimization. In *Proceedings of the 19th Annual Conference on Learning Theory*. Lecture Notes in Computer Science, vol. 4005. Springer, Berlin, 514–528.

WATROUS, J. 1999. PSPACE has constant-round quantum interactive proof systems. In *Proceedings of the 40th Annual IEEE Symposium on Foundations of Computer Science*. IEEE Computer Society Press, Los Alamitos, CA, 112–119.

WATROUS, J. 2009. Zero-knowledge against quantum attacks. *SIAM J. Comput. 39,* 1, 25–58.

Received March 2011; revised August 2011; accepted September 2011