14th Conference on the Theory of Quantum Computation, Communication and Cryptography

TQC 2019, June 3–5, 2019, Maryland, USA

Edited by

Wim van Dam

Laura Mančinska

Editors

Wim van Dam
University of California, Santa Barbara, CA, U.S.A.
QC Ware, Palo Alto, CA, U.S.A.
vandam@ucsb.edu

Laura Mančinska

University of Copenhagen, Denmark
mancinska@math.ku.dk

ACM Classification 2012
Theory of computation → Quantum computation theory; Theory of computation → Quantum complexity theory; Theory of computation → Quantum communication complexity; Theory of computation → Quantum query complexity; Theory of computation → Quantum information theory; Theory of computation → Quantum complexity theory; Hardware → Quantum communication and cryptography; Hardware → Quantum error correction and fault tolerance; Computer systems organization → Quantum computing

ISBN 978-3-95977-112-2

Published online and open access by
Schloss Dagstuhl – Leibniz-Zentrum für Informatik GmbH, Dagstuhl Publishing, Saarbrücken/Wadern, Germany. Online available at https://www.dagstuhl.de/dagpub/978-3-95977-112-2.

Publication date
May, 2019

Bibliographic information published by the Deutsche Nationalbibliothek
The Deutsche Nationalbibliothek lists this publication in the Deutsche Nationalbibliografie; detailed bibliographic data are available in the Internet at https://portal.dnb.de.

Digital Object Identifier: 10.4230/LIPIcs.TQC.2019.0

ISBN 978-3-95977-112-2 **ISSN 1868-8969** **https://www.dagstuhl.de/lipics**

LIPIcs – Leibniz International Proceedings in Informatics

LIPIcs is a series of high-quality conference proceedings across all fields in informatics. LIPIcs volumes are published according to the principle of Open Access, i.e., they are available online and free of charge.

ISSN 1868-8969

https://www.dagstuhl.de/lipics

Contents

Charter, Previous Editions, Steering Committee 0:vii

Organization TQC 2019 ..... 0:ix

Outstanding Paper Award 0:xi

Accepted Workshop Talks 0:xiii

Regular Papers

On Quantum Chosen-Ciphertext Attacks and Learning with Errors
Gorjan Alagic, Stacey Jeffery, Maris Ozols, and Alexander Poremba 1:1–1:23

Quantum Distinguishing Complexity, Zero-Error Algorithms, and Statistical Zero Knowledge
Shalev Ben-David and Robin Kothari 2:1–2:23

Circuit Transformations for Quantum Architectures
Andrew M. Childs, Eddie Schoute, and Cem M. Unsal 3:1–3:24

The RGB No-Signalling Game
Xavier Coiteux-Roy and Claude Crépeau 4:1–4:17

On the Qubit Routing Problem
Alexander Cowtan, Silas Dilkes, Ross Duncan, Alexandre Krajenbrink, Will Simmons, and Seyon Sivarajah 5:1–5:32

Applications of the Quantum Algorithm for st-Connectivity
Kai DeLorenzo, Shelby Kimmel, and R. Teal Witter 6:1–6:14

Bayesian ACRONYM Tuning
John Gamble, Christopher Granade, and Nathan Wiebe 7:1–7:19

A Compressed Classical Description of Quantum States
David Gosset and John Smolin 8:1–8:9

Approximate Unitary $n^{2/3}$-Designs Give Rise to Quantum Channels with Super Additive Classical Holevo Capacity
Aditya Nema and Pranab Sen 9:1–9:22

Parameterization of Tensor Network Contraction
Bryan O'Gorman 10:1–10:19

14th Conference on the Theory of Quantum Computation, Communication and Cryptography.
Editors: Wim van Dam and Laura Mančinska
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Charter, Previous Editions, Steering Committee

TQC Charter Statement

The Conference on the Theory of Quantum Computation, Communication and Cryptography (TQC) is a conference for students and researchers working on theoretical aspects of quantum computation and quantum information. This includes, but is not limited to, quantum algorithms, models of quantum computation, quantum complexity theory, simulation of quantum systems, quantum cryptography, quantum communication, quantum information theory, quantum estimation and measurement, the intersection of quantum information and condensed-matter theory, quantum coding theory, fault-tolerant quantum computing, and entanglement theory. It is the goal of the conference to present recent major results on the theory of quantum computing and to support the building of a research community.

Previous Editions of TQC

- 2018, July 16–18, University of Technology Sydney, Australia
- 2017, June 14–16, Université Pierre et Marie Curie, Paris, France
- 2016, September 27–29, Berlin, Germany
- 2015, May 20–22, Université libre de Bruxelles, Brussels, Belgium
- 2014, May 21–23, Centre for Quantum Technologies, National University of Singapore
- 2013, May 21–23, University of Guelph, Canada
- 2012, May 17–19, University of Tokyo, Japan
- 2011, May 24–26, Madrid, Spain
- 2010, April 13–15, University of Leeds, UK
- 2009, May 11–13, Institute for Quantum Computing, University of Waterloo, Canada
- 2008, January 30–February 1, University of Tokyo, Tokyo, Japan
- 2007, January 24–25, Nara Institute of Science and Technology, Nara, Japan
- 2006, February 22–23, NTT R&D Center, Atsugi, Kanagawa, Japan

TQC Steering Committee

- Anne Broadbent
- Wim van Dam
- Aram Harrow (chair)
- Stacey Jeffery
- Yasuhito Kawano
- Martin Roetteler
- Simone Severini
- Marco Tomamichel

14th Conference on the Theory of Quantum Computation, Communication and Cryptography.
Editors: Wim van Dam and Laura Mančinska

Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Organization TQC 2019

Program Committee

- Juan Miguel Arrazola
- Salman Beigi
- Mario Berta
- Sergey Bravyi
- André Chailloux
- Eric Chitambar
- Elizabeth Crosson
- Wim van Dam (co-chair)
- Omar Fawzi
- Steven Flammia
- Keisuke Fujii
- Ernesto F. Galvão
- Rahul Jain
- Zhengfeng Ji
- Shelby Kimmel
- Robert König
- François Le Gall
- Laura Mančinska (chair)
- Serge Massar
- Tobias J. Osborne
- Maris Ozols
- Christopher Portmann
- Ana Belén Sainz
- Pranab Sen
- Barbara Terhal
- John Watrous
- Ronald de Wolf

Local Organizing Committee

- Gorjan Alagic (chair)
- Alexey Gorshkov
- Andrew Childs
- Yi-Kai Liu
- Carl Miller
- Aarthi Sundaram (chair)
- Jacob Taylor

14th Conference on the Theory of Quantum Computation, Communication and Cryptography.
Editors: Wim van Dam and Laura Mančinska
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Outstanding Paper Award

From the conference track submissions, the Program Committee selected as the TQC 2019 Outstanding Paper:

D. Gosset and J. Smolin, "A compressed classical description of quantum states"

14th Conference on the Theory of Quantum Computation, Communication and Cryptography.
Editors: Wim van Dam and Laura Mančinska
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Accepted Workshop Talks

- Gorjan Alagic, Tommaso Gagliardoni, and Christian Majenz, "Unforgeable authentication and signing of quantum states"
- Alessandro Bisio and Paolo Perinotti, "Axiomatic theory of higher-order quantum computation"
- Paul Boes, Jens Eisert, Rodrigo Gallego, Markus Mueller, and Henrik Wilming, "Von Neumann entropy from unitarity", merged with
  Paul Boes, Henrik Wilming, Rodrigo Gallego, and Jens Eisert, "Catalytic quantum randomness"
- Johannes Borregaard, Hannes Pichler, Tim Schröder, Mikhail D. Lukin, Peter Lodahl, and Anders S. Sørensen, "One-way quantum repeater with minimal-resources"
- Andrew M. Childs, Aaron Ostrander, and Yuan Su, "Faster quantum simulation by randomization", merged with
  Andrew M. Childs and Yuan Su, "Nearly optimal lattice simulation by product formulas"
- Matthias Christandl, Angelo Lucia, Péter Vrana, and Albert H. Werner, "Tensor network representations from the geometry of entangled states"
- Patricia Contreras-Tejada, Carlos Palazuelos, and Julio I. de Vicente, "A resource theory of entanglement with a unique multipartite maximally entangled state"
- Nilanjana Datta, Christoph Hirche, and Andreas Winter, "Convexity and operational interpretation of the quantum information bottleneck function"
- Philippe Faist, Sepehr Nezami, Victor V. Albert, Grant Salton, Fernando Pastawski, Patrick Hayden, and John Preskill, "Continuous symmetries and approximate quantum error correction", merged with
  Mischa P. Woods and Álvaro M. Alhambra, "Continuous groups of transversal gates for quantum error correcting codes from finite clock reference frames"
- Steven T. Flammia and Joel J. Wallman, "Efficient learning of Pauli channels"
- Carlos E. González-Guillén and Toby S. Cubitt, "History-state Hamiltonians are critical"
- Alex B. Grilo, " A simple protocol for verifiable delegation of quantum computation in one round"
- Arne L. Grimsmo, Joshua Combes, and Ben Q. Baragiola, "Quantum computing with rotation-symmetric bosonic codes"
- Tamara Kohler and Toby Cubitt, "Toy models of holographic duality between local Hamiltonians"
- François Le Gall, Harumichi Nishimura, and Ansis Rosmanis, "Quantum advantage for the LOCAL model in distributed computing"
- Guang Hao Low, Vadym Kliuchnikov, and Luke Schaeffer, "Trading T-gates for dirty qubits in state preparation and unitary synthesis"
- Iman Marvian and Seth Lloyd, "Universal quantum emulator"
- Thomas Vidick and Tina Zhang, "Classical zero-knowledge arguments for quantum computations"
- Guanyu Zhu, Ali Lavasani, and Maissam Barkeshli, "Universal logical gate sets with constant-depth circuits for topological and hyperbolic quantum codes"

14th Conference on the Theory of Quantum Computation, Communication and Cryptography.
Editors: Wim van Dam and Laura Mančinska

Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

On Quantum Chosen-Ciphertext Attacks and Learning with Errors

Gorjan Alagic
QuICS, University of Maryland, MD, USA
NIST, Gaithersburg, MD, USA
galagic@umd.edu

Stacey Jeffery
QuSoft, Amsterdam, Netherlands
CWI, Amsterdam, Netherlands
smjeffery@gmail.com

Maris Ozols
QuSoft, Amsterdam, Netherlands
University of Amsterdam, Netherlands
marozols@gmail.com

Alexander Poremba
Computing and Mathematical Sciences, Caltech, Pasadena, CA, USA
aporemba@caltech.edu

Abstract

Quantum computing is a significant threat to classical public-key cryptography. In strong "quantum access" security models, numerous symmetric-key cryptosystems are also vulnerable. We consider classical encryption in a model which grants the adversary quantum oracle access to encryption and decryption, but where the latter is restricted to non-adaptive (i.e., pre-challenge) queries only. We define this model formally using appropriate notions of ciphertext indistinguishability and semantic security (which are equivalent by standard arguments) and call it QCCA1 in analogy to the classical CCA1 security model. Using a bound on quantum random-access codes, we show that the standard PRF-based encryption schemes are QCCA1-secure when instantiated with quantum-secure primitives.

We then revisit standard IND-CPA-secure Learning with Errors (LWE) encryption and show that leaking just one quantum decryption query (and no other queries or leakage of any kind) allows the adversary to recover the full secret key with constant success probability. In the classical setting, by contrast, recovering the key requires a linear number of decryption queries. The algorithm at the core of our attack is a (large-modulus version of) the well-known Bernstein-Vazirani algorithm. We emphasize that our results should **not** be interpreted as a weakness of these cryptosystems in their stated security setting (i.e., post-quantum chosen-plaintext secrecy). Rather, our results mean that, if these cryptosystems are exposed to chosen-ciphertext attacks (e.g., as a result of deployment in an inappropriate real-world setting) then quantum attacks are even more devastating than classical ones.

2012 ACM Subject Classification Theory of computation → Quantum computation theory; Security and privacy → Cryptanalysis and other attacks

Keywords and phrases quantum chosen-ciphertext security, quantum attacks, learning with errors

Digital Object Identifier 10.4230/LIPIcs.TQC.2019.1

Related Version A full version of the paper is available at `https://eprint.iacr.org/2018/1185`.

Funding *Gorjan Alagic*: Supported by National Science Foundation (NSF) grant CCF-1763736.
Stacey Jeffery: Supported by an NWO WISE Grant and NWO Veni Innovational Research Grant under project number 639.021.752.
Maris Ozols: Supported by Leverhulme Trust Early Career Fellowship (ECF-2015-256).
Alexander Poremba: Partial support from AFOSR YIP award number FA9550-16-1-0495, the IQIM – an NSF Physics Frontiers Center (NSF Grant PHY- 1733907), and the Kortschak Scholars program.

Acknowledgements We thank Ronald de Wolf for helpful discussions and Jop Briët for Lemma 25.

1 Introduction

Large-scale quantum computers pose a dramatic threat to classical cryptography. The ability of such devices to run Shor's efficient quantum factoring algorithm (and its variants) would lead to devastation of the currently deployed public-key cryptography infrastructure [8, 25]. This threat has led to significant work on so-called "post-quantum" alternatives, where a prominent category is occupied by cryptosystems based on the *Learning with Errors* (LWE) problem of solving noisy linear equations over $\mathbb{Z}_q$ [23] and its variants [8, 22].

In addition to motivating significant work on post-quantum cryptosystems, the threat of quantum computers has spurred general research on secure classical cryptography in the presence of quantum adversaries. One area in particular explores security models where a quantum adversary gains quantum control over portions of a classical cryptosystem. In such models, a number of basic symmetric-key primitives can be broken by simple quantum attacks based on Simon's algorithm [16, 17, 15, 24, 26]. It is unclear if the assumption behind these models is plausible for typical physical implementations of symmetric-key cryptography. However, attacks that involve quantumly querying a classical function are always available in scenarios where the adversary has access to a circuit for the relevant function. This is the case for hashing, public-key encryption, and circuit obfuscation. Moreover, understanding this model is crucial for gauging the degree to which any physical cryptographic device must be resistant to reverse engineering or forced quantum behavior (consider the so-called "frozen smart card" example [10]). For instance, one may reasonably ask: *what happens to the security of a classical cryptosystem when the device leaks only a single quantum query to the adversary?*

When deciding which functions the adversary might have (quantum) access to, it is worth recalling the classical setting. For classical symmetric-key encryption, a standard approach considers the security of cryptosystems when exposed to so-called chosen-plaintext attacks (CPA). This notion encompasses all attacks in which an adversary attempts to defeat security (by, e.g., distinguishing ciphertexts or extracting key information) using oracle access to the function which encrypts plaintexts with the secret key. This approach has been highly successful in developing cryptosystems secure against a wide range of realistic real-world attacks. An analogous class, the so-called chosen-ciphertext attacks (CCA), are attacks in which the adversary can make use of oracle access to decryption. For example, a well-known attack due to Bleichenbacher [4] only requires access to an oracle that decides if the input ciphertext is encrypted according to a particular RSA standard. We will consider analogues of both CPA and CCA attacks, in which the relevant functions are quantumly accessible to the adversary.

Prior works have formalized the quantum-accessible model for classical cryptography in several settings, including unforgeable message authentication codes and digital signatures [6, 5], encryption secure against quantum chosen-plaintext attacks (QCPA) [7, 10], and encryption secure against *adaptive* quantum chosen-ciphertext attacks (QCCA2) [6].

1.1 Our Contributions

1.1.1 The model

In this work, we define a quantum-secure model of encryption called QCCA1. This model grants *non-adaptive* access to the decryption oracle, and is thus intermediate between QCPA and QCCA2. Studying weaker and intermediate models is a standard and useful practice in cryptography. In fact, CPA and CCA2 are intermediate models themselves, being strictly

weaker than authenticated encryption. Our particular intermediate model is naturally motivated: it is sufficent for a new and interesting quantum attack on LWE encryption.

As is typical, the "challenge" in QCCA1 can be semantic, or take the form of an indistinguishability test. This leads to natural security notions for symmetric-key encryption, which we call IND-QCCA1 and SEM-QCCA1, respectively. Following previous works, it is straightforward to prove that IND-QCCA1 and SEM-QCCA1 are equivalent [7, 10, 6].

We prove IND-QCCA1 security for two symmetric-key encryption schemes, based on standard assumptions. Specifically, we show that the standard encryption scheme based on quantum-secure pseudorandom functions (QPRF) is IND-QCCA1-secure. We remark that QPRFs can be constructed from quantum-secure one-way functions [28]. Our security proofs use a novel technique, in which we control the amount of information that the adversary can extract from the oracles and store in their internal quantum state (prior to the challenge) by means of a certain bound on quantum random-access codes.

1.1.2 A quantum-query attack on LWE

We then revisit the aforementioned question: what happens to a post-quantum cryptosystem if it leaks a single quantum query? Our main result is that standard IND-CPA-secure LWE-based encryption schemes can be completely broken using only *a single quantum decryption query* and no other queries or leakage of any kind. In our attack, the adversary recovers the complete secret key with constant success probability. In standard bit-by-bit LWE encryption, a single classical decryption query can yield at most one bit of the secret key; the classical analogue of our attack thus requires $n \log q$ queries. The attack is essentially an application of a modulo-q variant of the Bernstein-Vazirani algorithm [3]. Our new analysis shows that this algorithm correctly recovers the key with constant success probability, despite the decryption function only returning an inner product which is rounded to one of two values. We show that the attack applies to four variants of standard IND-CPA-secure LWE-based encryption: the symmetric-key and public-key systems originally described by Regev [23], the FrodoPKE scheme[1] [18, 1], and standard Ring-LWE [19, 20].

1.1.3 Important caveats

Our results challenge the idea that LWE is unconditionally "just as secure" quantumly as it is classically. Nonetheless, the reader is cautioned to interpret our work carefully. Our results do *not* indicate a weakness in LWE (or any LWE-based cryptosystem) in the standard post-quantum security model. Since it is widely believed that quantum-algorithmic attacks will need to be launched over purely classical channels, post-quantum security does not allow for quantum queries to encryption or decryption oracles. Moreover, while our attack does offer a dramatic quantum speedup (i.e., one query vs. linear queries), the classical attack is already efficient. The schemes we attack are already insecure in the classical chosen-ciphertext setting, but can be modified to achieve chosen-ciphertext security [9].

[1] FrodoPKE is an IND-CPA-secure building block in the IND-CCA2-secure post-quantum cryptosystem "FrodoKEM" [1]. Our results do not affect the post-quantum security of Frodo and do not contradict the CCA2 security of FrodoKEM.

1.1.4 Related work

We remark that Grilo, Kerenidis and Zijlstra recently observed that a version of LWE with so-called "quantum samples" can be solved efficiently (as a learning problem) using Bernstein-Vazirani [14]. Our result, by contrast, demonstrates an actual cryptographic attack on standard cryptosystems based on LWE, in a plausible security setting. Moreover, in terms of solving the learning problem, our analysis shows that constant success probability is achievable with only a single query, whereas [14] require a number of queries which is at least linear in the modulus q. In particular, our cryptographic attack succeeds with a single query even for superpolynomial modulus.

1.2 Technical summary of results

1.2.1 Security model and basic definitions

We set down the basic QCCA1 security model, adapting the ideas of [5, 10]. An encryption scheme is a triple $\Pi = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ of algorithms (key generation, encryption, and decryption, respectively) satisfying $\mathsf{Dec}_k(\mathsf{Enc}_k(m)) = m$ for any key $k \leftarrow \mathsf{KeyGen}$ and message m. In what follows, all oracles are quantum, meaning a function f is accessed by the unitary $|x\rangle|y\rangle \mapsto |x\rangle|y \oplus f(x)\rangle$. We define indistinguishability and semantic security as follows.

▶ **Definition 1** (informal)**.** Π *is* IND-QCCA1 *if no quantum polynomial-time algorithm (*QPT*)* $\mathcal{A}$ *can succeed at the following experiment with probability better than* $1/2 + \mathrm{negl}(n)$*.*

1. *A key* $k \leftarrow \mathsf{KeyGen}(1^n)$ *and a uniformly random bit* $b \xleftarrow{\$} \{0,1\}$ *are generated;* $\mathcal{A}$ *gets access to oracles* Enc_k *and* Dec_k*, and outputs* (m_0, m_1)*;*
2. $\mathcal{A}$ *gets* $\mathsf{Enc}_k(m_b)$ *and access to an oracle for* Enc_k*, and outputs a bit* b'*;* $\mathcal{A}$ *wins if* $b = b'$*.*

▶ **Definition 2** (informal)**.** *Consider the following game with a* QPT $\mathcal{A}$*.*

1. *A key* $k \leftarrow \mathsf{KeyGen}(1^n)$ *is generated;* $\mathcal{A}$ *gets access to oracles* Enc_k*,* Dec_k *and outputs circuits* (Samp, h, f)*;*
2. *Sample* $m \leftarrow \mathsf{Samp}$*;* $\mathcal{A}$ *receives* $h(m)$*,* $\mathsf{Enc}_k(m)$*, and access to an oracle for* Enc_k *only, and outputs a string* s*;* $\mathcal{A}$ *wins if* $s = f(m)$*.*

Then Π *is* SEM-QCCA1 *if for every* QPT $\mathcal{A}$ *there exists a* QPT $\mathcal{S}$ *with the same winning probability but which does not get* $\mathsf{Enc}_k(m)$ *in step 2.*

▶ **Theorem 3.** *A symmetric-key encryption scheme is* IND-QCCA1 *if and only if it is* SEM-QCCA1*.*

1.2.2 Secure constructions

Next, we show that standard pseudorandom-function-based encryption is QCCA1-secure, provided that the underlying PRF is quantum-secure (i.e., is a QPRF.) A QPRF can be constructed from any quantum-secure one-way function, or directly from the LWE assumption [28]. Given a PRF $f = \{f_k\}_k$, define $\mathsf{PRFscheme}[f]$ to be the scheme which encrypts a plaintext m using randomness r via $\mathsf{Enc}_k(m; r) = (r, f_k(r) \oplus m)$ and decrypts in the obvious way.

▶ **Theorem 4.** *If* f *is a* QPRF*, then* $\mathsf{PRFscheme}[f]$ *is* IND-QCCA1*-secure.*

In the full version of this work, we also analyze a standard permutation-based scheme. Quantum-secure PRPs (i.e., QPRPs) can be obtained from quantum-secure one-way functions [29]. Given a PRP $P = \{P_k\}_k$, define $\mathsf{PRPscheme}[P]$ to be the scheme that encrypts a

plaintext m using randomness r via $\mathsf{Enc}_k(m;r) = P_k(m||r)$, where $||$ denotes concatenation; to decrypt, apply P_k^{-1} and discard r.

▶ **Theorem 5.** *If P is a* QPRP*, then* PRPscheme$[P]$ *is* IND-QCCA1*-secure.*

We briefly describe our proof techniques for Theorems 4 and 5. In the indistinguishability game, the adversary can use the decryption oracle prior to the challenge to (quantumly) encode information about the relevant pseudorandom function instance (i.e., f_k or P_k) in his private, poly-sized quantum memory. To establish security, it is enough to show that this encoded information cannot help the adversary compute the value of the relevant function at the particular randomness used in the challenge. To prove this, we use a bound on quantum random access codes (QRAC), where, informally, a QRAC is a mapping from N-bit strings x to d-dimensional states ϱ_x, such that given ϱ_x, and any $j \in [N]$, x_j can be recovered with some probability.

▶ **Lemma 6.** *The average bias of a quantum random access code with shared randomness that encodes N bits into a d-dimensional quantum state is $O(\sqrt{N^{-1}\log d})$. In particular, if $N = 2^n$ and $d = 2^{\mathrm{poly}(n)}$ the bias is $O(2^{-n/2}\,\mathrm{poly}(n))$.*

1.2.3 Key recovery against LWE

Our attack on LWE encryption uses a new analysis of a large-modulus variant of the Bernstein-Vazirani algorithm [3], in the presence of a certain type of "rounding" noise.

1.2.4 Quantum algorithm for linear rounding functions

Given integers $n \geq 1$ and $q \geq 2$, define a keyed family of (binary) linear rounding functions, $\mathsf{LRF}_{\boldsymbol{k},q} : \mathbb{Z}_q^n \longrightarrow \{0,1\}$, with key $\boldsymbol{k} \in \mathbb{Z}_q^n$, as follows:

$$\mathsf{LRF}_{\boldsymbol{k},q}(\boldsymbol{x}) := \begin{cases} 0 & \text{if } |\langle \boldsymbol{x}, \boldsymbol{k}\rangle| \leq \lfloor \frac{q}{4} \rfloor, \\ 1 & \text{otherwise.} \end{cases}$$

Here $\langle \cdot, \cdot \rangle$ denote the inner product modulo q. Our main technical contribution is the following.

▶ **Theorem 7** (informal)**.** *There exists a quantum algorithm that runs in time $O(n)$, makes one quantum query to $\mathsf{LRF}_{\boldsymbol{k},q}$ (with unknown $\boldsymbol{k} \in \mathbb{Z}_q^n$), and outputs $\boldsymbol{k}$ with probability $4/\pi^2 - O(1/q)$.*

We also show that the same algorithm succeeds against more generalized function classes, in which the oracle indicates which "segment" of $\mathbb{Z}_q$ the exact inner product belongs to.

1.2.5 One quantum query against LWE

Finally, we revisit our central question of interest: what happens to a post-quantum cryptosystem if it leaks a single quantum query? We show that, in standard LWE-based schemes, the decryption function can (with some simple modifications) be viewed as a special case of a linear rounding function, as above. In standard symmetric-key or public-key LWE, for instance, we decrypt a ciphertext $(\boldsymbol{a}, c) \in \mathbb{Z}_q^{n+1}$ with key $\boldsymbol{k}$ by outputting 0 if $|c - \langle \boldsymbol{a}, \boldsymbol{k}\rangle| \leq \lfloor \frac{q}{4} \rfloor$ and 1 otherwise. In standard Ring-LWE, we decrypt a ciphertext (u, v) with key k (here u, v, k are polynomials in $\mathbb{Z}_q[x]/\langle x^n + 1\rangle$) by outputting 0 if the constant coefficient of $v - k \cdot u$ is small, and 1 otherwise.

Each of these schemes is secure against adversaries with classical encryption oracle access, under the LWE assumption. If adversaries also gain classical decryption access, then it's not hard to see that a linear number of queries is necessary and sufficient to recover the private key. Our main result is that, by contrast, only a *single* quantum decryption query is required to achieve this total break. Indeed, in all three constructions described above, one can use the decryption oracle to build an associated oracle for a linear rounding function which hides the secret key. The following can then be shown using Theorem 7.

▶ **Theorem 8** (informal). *Let* Π *be standard* LWE *or standard* Ring-LWE *encryption (either symmetric-key, or public-key.) Let* n *be the security parameter. Then there is an efficient quantum algorithm that runs in time* $O(n)$*, uses one quantum query to the decryption function* $\mathsf{Dec}_{\boldsymbol{k}}$ *of* Π*,and outputs the secret key with constant probability.*

It's natural to ask whether a similar quantum speedup can be achieved using encryption queries. In the full version of this article, we show that this is possible in a model in which the adversary is allowed to select some of the random coins used to encrypt.

1.3 Organization

The remainder of this paper is organized as follows. In Section 2, we outline preliminary ideas that we will make use of, including cryptographic concepts, and notions from quantum algorithms. In Section 3, we define the QCCA1 model, including the two equivalent versions IND-QCCA1 and SEM-QCCA1. In Section 4, we define the PRF scheme, and show that they are IND-QCCA1-secure. In Section 5, we show how a generalization of the Bernstein-Vazirani algorithm works with probability bounded from below by a constant, even when the oracle outputs rounded values. In Section 6, we use the results of Section 5 to prove that a single quantum decryption query is enough to recover the secret key in various versions of LWE-encryption.

2 Preliminaries

2.1 Basic notation and conventions

Selecting an element x uniformly at random from a finite set X will be written as $x \xleftarrow{\$} X$. If we are generating a vector or matrix with entries in $\mathbb{Z}_q$ by sampling each entry independently according to a distribution χ on $\mathbb{Z}_q$, we will write, e.g., $\boldsymbol{v} \xleftarrow{\chi} \mathbb{Z}_q^n$. Given a matrix A, A^{T} will denote the transpose of A. We will view elements $\boldsymbol{v}$ of $\mathbb{Z}_q^n$ as column vectors; the notation $\boldsymbol{v}^{\mathsf{T}}$ then denotes the corresponding row vector. The notation $\mathrm{negl}(n)$ denotes some function of n which is smaller than every inverse-polynomial. We denote the concatenation of strings x and y by $x||y$. We abbreviate classical probabilistic polynomial-time algorithms as PPT algorithms. By *quantum algorithm* (or QPT) we mean a polynomial-time uniform family of quantum circuits, where each circuit in the family is described by a sequence of unitary gates and measurements. In general, such an algorithm may receive (mixed) quantum states as inputs and produce (mixed) quantum states as outputs. Sometimes we will restrict QPTs implicitly; for example, if we write $\Pr[\mathcal{A}(1^n) = 1]$ for a QPT $\mathcal{A}$, it is implicit that we are only considering those QPTs that output a single classical bit.

A function $f : \{0,1\}^m \to \{0,1\}^\ell$ defines a unitary operator $U_f : |x\rangle|y\rangle \to |x\rangle|y \oplus f(x)\rangle$ on $m + \ell$ qubits where $x \in \{0,1\}^m$ and $y \in \{0,1\}^\ell$. When we say that a quantum algorithm $\mathcal{A}$ gets (adaptive) quantum oracle access to f (written $\mathcal{A}^f$), we mean that $\mathcal{A}$ can apply U_f. Recall that a symmetric-key encryption scheme is a triple of classical probabilistic algorithms

(KeyGen, Enc, Dec) whose run-times are polynomial in some security parameter n. Such a scheme must satisfy the following property: when a key k is sampled by running $\mathsf{KeyGen}(1^n)$, then it holds that $\mathsf{Dec}_k(\mathsf{Enc}_k(m)) = m$ for all m except with negligible probability in n. In this work, all encryption schemes will be fixed-length, i.e., the length of the message m will be a fixed (at most polynomial) function of n.

Since the security notions we study are unachievable in the information-theoretic setting, all adversaries will be modeled by QPTs. When security experiments require multiple rounds of interaction with the adversary, $\mathcal{A}$ is implicitly split into multiple QPTs (one for each round), and each algorithm forwards its internal (quantum) state to the next algorithm in the sequence.

2.2 Quantum-secure pseudorandomness

Let $f : \{0,1\}^n \times \{0,1\}^m \to \{0,1\}^\ell$ be an efficiently computable function, where n, m, ℓ are integers and where f defines a family of functions $\{f_k\}_{k\in\{0,1\}^n}$ with $f_k(x) = f(k,x)$. We say f is a *quantum-secure pseudorandom function* (or QPRF) if, for every QPT $\mathcal{A}$,

$$\left| \Pr_{k \xleftarrow{\$} \{0,1\}^n} \left[\mathcal{A}^{f_k}(1^n) = 1\right] - \Pr_{g \xleftarrow{\$} \mathcal{F}_m^\ell} \left[\mathcal{A}^{g}(1^n) = 1\right] \right| \leq \mathrm{negl}(n)\,. \tag{1}$$

Here $\mathcal{F}_m^\ell$ denotes the set of all functions from $\{0,1\}^m$ to $\{0,1\}^\ell$. The standard method for constructing a pseudorandom function from a one-way function produces a QPRF, provided that the one-way function is quantum-secure [13, 12, 28].

2.3 Quantum random access codes

A *quantum random access code* (QRAC) is a two-party scheme for the following scenario involving two parties Alice and Bob [21]:

1. Alice gets $x \in \{0,1\}^N$ and encodes it as a d-dimensional quantum state ϱ_x.
2. Bob receives ϱ_x from Alice, and some index $i \in \{1,\ldots,N\}$, and is asked to recover the i-th bit of x, by performing some measurement on ϱ_x.
3. They win if Bob's output agrees with x_i and lose otherwise.

We can view a QRAC scheme as a pair of (not necessarily efficient) quantum algorithms: one for encoding, and another for decoding. We remark that the definition of a QRAC does not bound on the size of ϱ_x; the interesting question is with what parameters a QRAC can actually exist.

A variation of the above scenario allows Alice and Bob to use *shared randomness* in their encoding and decoding operations [2]. Hence, Alice and Bob can pursue probabilistic strategies with access to the same random variable.

Define the average bias of a QRAC with shared randomness as $\epsilon = p_{\mathrm{win}} - 1/2$, where p_{win} is the winning probability averaged over $x \xleftarrow{\$} \{0,1\}^N$ and $i \xleftarrow{\$} \{1,\ldots,N\}$.

3 The QCCA1 security model

3.1 Quantum oracles

In our setting, adversaries will (at various times) have quantum oracle access to the classical functions Enc_k and Dec_k. The case of the deterministic decryption function Dec_k is simple: the adversary gets access to the unitary operator $U_{\mathsf{Dec}_k} : |c\rangle|m\rangle \mapsto |c\rangle|m \oplus \mathsf{Dec}_k(c)\rangle$. For encryption, to satisfy IND-CPA security, Enc_k must be probabilistic and thus does not

correspond to any single unitary operator. Instead, each encryption oracle call of the adversary will be answered by applying a unitary sampled uniformly from the family $\{U_{\mathsf{Enc}_k,r}\}_r$ where

$$U_{\mathsf{Enc}_k,r} : |m\rangle|c\rangle \mapsto |m\rangle|c \oplus \mathsf{Enc}_k(m;r)\rangle$$

and r varies over all possible values of the randomness register of Enc_k. Note that, since Enc_k and Dec_k are required to be probabilistic polynomial-time algorithms provided by the underlying classical symmetric-key encryption scheme, both $U_{\mathsf{Enc}_k,r}$ and U_{Dec_k} correspond to efficient and reversible quantum operations. For the sake of brevity, we adopt the convenient notation Enc_k and Dec_k to refer to the above quantum oracles for encryption and decryption respectively.

3.2 Ciphertext indistinguishability

We now define indistinguishability of encryptions (for classical, symmetric-key schemes) against non-adaptive quantum chosen-ciphertext attacks.

▶ **Definition 9** (IND-QCCA1). *Let* $\Pi = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ *be an encryption scheme,* $\mathcal{A}$ *a* QPT*, and* n *the security parameter. Define* $\mathsf{IndGame}(\Pi, \mathcal{A}, n)$ *as follows.*

1. Setup: *A key* $k \leftarrow \mathsf{KeyGen}(1^n)$ *and a bit* $b \xleftarrow{\$} \{0,1\}$ *are generated;*
2. Pre-challenge: $\mathcal{A}$ *gets access to oracles* Enc_k *and* Dec_k*, and outputs* (m_0, m_1)*;*
3. Challenge: $\mathcal{A}$ *gets* $\mathsf{Enc}_k(m_b)$ *and access to* Enc_k *only, and outputs a bit* b'*;*
4. Resolution: $\mathcal{A}$ *wins if* $b = b'$*.*

Then Π *has indistinguishable encryptions under non-adaptive quantum chosen ciphertext attack (or is* IND-QCCA1*) if, for every* QPT $\mathcal{A}$*,*

$$\Pr[\mathcal{A} \text{ wins } \mathsf{IndGame}(\Pi, \mathcal{A}, n)] \leq 1/2 + \mathrm{negl}(n)\,.$$

By inspection, one immediately sees that our definition lies between the established notions of IND-QCPA and IND-QCCA2 [7, 10, 6]. It will later be convenient to work with a variant of the game IndGame, which we now define.

▶ **Definition 10** (IndGame′). *We define the experiment* $\mathsf{IndGame}'(\Pi, \mathcal{A}, n)$ *as* $\mathsf{IndGame}(\Pi, \mathcal{A}, n)$*, except that in the pre-challenge phase* $\mathcal{A}$ *only outputs a single message* m*, and in the challenge phase* $\mathcal{A}$ *receives* $\mathsf{Enc}_k(m)$ *if* $b = 0$*, and* $\mathsf{Enc}_k(x)$ *for a uniformly random message* x *if* $b = 1$*.*

Working with $\mathsf{IndGame}'$ rather than IndGame does not change security. Specifically (as we show in Appendix B), Π is IND-QCCA1 if and only if, for every QPT $\mathcal{A}$,

$$\Pr[\mathcal{A} \text{ wins } \mathsf{IndGame}'(\Pi, \mathcal{A}, n)] \leq 1/2 + \mathrm{negl}(n)\,.$$

3.3 Semantic security

In semantic security, rather than choosing a pair of challenge plaintexts, the adversary chooses a *challenge template*: a triple of circuits (Samp, h, f), where Samp outputs plaintexts from some distribution $\mathcal{D}_{\mathsf{Samp}}$, and h and f are functions with domain the support of $\mathcal{D}_{\mathsf{Samp}}$. The intuition is that Samp is a distribution of plaintexts m for which the adversary, if given information $h(m)$ about m together with an encryption of m, can produce some new information $f(m)$.

▶ **Definition 11** (SEM-QCCA1). *Let* $\Pi = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ *be an encryption scheme, and consider the following experiment,* $\mathsf{SemGame}(b)$*, (with parameter* $b \in \{\mathsf{real}, \mathsf{sim}\}$*) with a* QPT $\mathcal{A}$*:*

1. Setup: *A key* $k \leftarrow \mathsf{KeyGen}(1^n)$ *is generated;*
2. Pre-challenge: $\mathcal{A}$ *gets access to oracles* Enc_k *and* Dec_k*, and outputs a challenge template* (Samp, h, f)*;*
3. Challenge: *A plaintext* $m \xleftarrow{\$} \mathsf{Samp}$ *is generated;* $\mathcal{A}$ *receives* $h(m)$ *and gets access to an oracle for* Enc_k *only; if* $b = \mathsf{real}$*,* $\mathcal{A}$ *also receives* $\mathsf{Enc}_k(m)$*;* $\mathcal{A}$ *outputs a string* s*;*
4. Resolution: $\mathcal{A}$ *wins if* $s = f(m)$*.*

We say Π *has semantic security under non-adaptive quantum chosen-ciphertext attack (or is* SEM-QCCA1*) if, for every* QPT $\mathcal{A}$*, there exists a* QPT $\mathcal{S}$ *such that the challenge templates output by* $\mathcal{A}$ *and* $\mathcal{S}$ *are identically distributed, and*

$$\left|\Pr[\mathcal{A} \text{ wins } \mathsf{SemGame}(\mathsf{real})] - \Pr[\mathcal{S} \text{ wins } \mathsf{SemGame}(\mathsf{sim})]\right| \leq \mathrm{negl}(n)\,.$$

Our definition is a straightforward modification of SEM-QCPA [10, 6]; the modification is to give $\mathcal{A}$ and $\mathcal{S}$ oracle access to Dec_k in the pre-challenge phase.

▶ **Theorem 12.** *Let* $\Pi = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ *be a symmetric-key encryption scheme. Then,* Π *is* IND-QCCA1*-secure if and only if* Π *is* SEM-QCCA1*-secure.*

The classical proof of the above (see, e.g., [11]) carries over to the quantum case. This was already observed for the case of QCPA by [10], and extends easily to the case where both the adversary and the simulator gain oracle access to Dec_k in the pre-challenge phase.[2]

4 Secure Constructions

4.1 PRF scheme

Let us first recall the standard symmetric-key encryption based on pseudorandom functions.

▶ **Construction 13** (PRF scheme)**.** *Let* n *be the security parameter and let* $f : \{0,1\}^n \times \{0,1\}^n \longrightarrow \{0,1\}^n$ *be an efficient family of functions* $\{f_k\}_k$*. Then, the symmetric-key encryption scheme* $\mathsf{PRFscheme}[f] = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ *is defined as follows:*

1. KeyGen: *output* $k \xleftarrow{\$} \{0,1\}^n$*;*
2. Enc: *to encrypt* $m \in \{0,1\}^n$*, choose* $r \xleftarrow{\$} \{0,1\}^n$ *and output* $(r, f_k(r) \oplus m)$*;*
3. Dec: *to decrypt* $(r, c) \in \{0,1\}^n \times \{0,1\}^n$*, output* $c \oplus f_k(r)$*;*

We chose a simple set of parameters for the PRF, so that key length, input size, and output size are all equal to the security parameter. It is straightforward to check that the definition (and our results) are valid for any polynomial-size parameter choices. We show that the above scheme satisfies QCCA1, provided that the underlying PRF is secure against quantum queries.

▶ **Theorem 14.** *If* f *is a* QPRF*, then* $\mathsf{PRFscheme}[f]$ *is* IND-QCCA1*-secure.*

Proof. Fix a QPT adversary $\mathcal{A}$, split into pre-challenge algorithm $\mathcal{A}_1$ and challenge algorithm $\mathcal{A}_2$, against $\Pi := \mathsf{PRFscheme}[f] = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$. Let n denote the security parameter.

We will work with the single-message variant of IndGame, IndGame′, described below as Game 0. In Appendix B, we show that Π is IND-QCCA1 if and only if no QPT adversary can win IndGame′ with non-negligible bias. We first show that a version of IndGame′ where we

[2] In fact, the proof works even if Dec_k access is maintained during the challenge, so the result is really that IND-QCCA2 is equivalent to SEM-QCCA2.

replace f with a random function, called Game 1 below, is indistinguishable from $\mathsf{IndGame}'$, so that the winning probabilities cannot differ by a non-negligible amount. We then prove that no adversary can win Game 1 with non-negligible bias by showing how any adversary for Game 1 can be used to make a quantum random access code with the same bias.

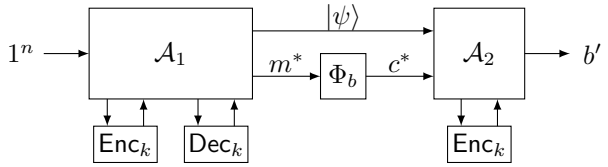


Figure 1 $\mathsf{IndGame}'$ from Definition 10.

Game 0. This is the game $\mathsf{IndGame}'(\Pi, \mathcal{A}, n)$, which we briefly review for convenience (see also Figure 1). In the pre-challenge phase, $\mathcal{A}_1$ gets access to oracles Enc_k and Dec_k, and outputs a message m^* while keeping a private state $|\psi\rangle$ for the challenge phase. In the challenge phase, a random bit $b \xleftarrow{\$} \{0,1\}$ is sampled, and $\mathcal{A}_2$ is run on input $|\psi\rangle$ and a challenge ciphertext

$$c^* := \Phi_b(m^*) := \begin{cases} \mathsf{Enc}_k(m^*) & \text{if } b = 0, \\ \mathsf{Enc}_k(x) & \text{if } b = 1. \end{cases}$$

Here $\mathsf{Enc}_k(x) := (r^*, f_k(r^*) \oplus x)$ where r^* and x are sampled uniformly at random. In the challenge phase, $\mathcal{A}_2$ only has access to Enc_k and must output a bit b'. $\mathcal{A}$ wins if $\delta_{bb'} = 1$, so we call $\delta_{bb'}$ the outcome of the game.

Game 1. This is the same game as Game 0, except we replace f_k with a uniformly random function $F : \{0,1\}^n \to \{0,1\}^n$.

First, we show that for any adversary $\mathcal{A}$, the outcome when $\mathcal{A}$ plays Game 0 is at most negligibly different from the outcome when $\mathcal{A}$ plays Game 1. We do this by constructing a quantum distinguisher $\mathcal{D}$ that distinguishes QPRF $\{f_k\}_k$ from a true random function, with advantage $\big|\Pr[1 \leftarrow \text{Game } 0] - \Pr[1 \leftarrow \text{Game } 1]\big|$, which must then be negligible since f is a QPRF. The distinguisher $\mathcal{D}$ gets quantum oracle access to a function g, which is either f_k, for a random k, or a random function, and proceeds by simulating $\mathcal{A}$ playing $\mathsf{IndGame}'$ as follows:

1. Run $\mathcal{A}_1$, answering encryption queries using calls to g instead of f_k, and decryption queries using quantum oracle calls to g: $|r\rangle|c\rangle|m\rangle \mapsto |r\rangle|c\rangle|m \oplus c\rangle \mapsto |r\rangle|c\rangle|m \oplus c \oplus g(r)\rangle$;
2. Simulate the challenge phase by sampling $b \xleftarrow{\$} \{0,1\}$ and encrypting the challenge using g in place of f_k; run $\mathcal{A}_2$ and simulate encryption queries as before;
3. When $\mathcal{A}_2$ outputs b', output $\delta_{bb'}$.

To show that no QPT adversary can win Game 1 with non-negligible probability, we design a QRAC from any adversary, and use the lower bound on the bias given in Lemma 6.

Intuition: In an encryption query, $\mathcal{A}_1$ or $\mathcal{A}_2$, queries a message, or superposition of messages $\sum_m |m\rangle$, and gets back $\sum_m |m\rangle|r, m \oplus F(r)\rangle$ for a random r, from which he can easily obtain $(r, F(r))$. So in essence, an encryption query is just classically sampling a random point of F.

In a decryption query, which is only available to $\mathcal{A}_1$, the adversary sends a ciphertext, or a superposition of ciphertexts, $\sum_{r,c} |r, c\rangle$ and gets back $\sum_{r,c} |r, c\rangle|c \oplus F(r)\rangle$, from which he can learn $\sum_r |r, F(r)\rangle$. Thus, a decryption query allows $\mathcal{A}_1$ to query F, in superposition.

Later in the challenge phase, $\mathcal{A}_2$ gets an encryption $(r^*, m \oplus F(r^*))$ and must decide if $m = m^*$. Since $\mathcal{A}_2$ no longer has access to the decryption oracle, which allows him to query F, there seem to be two possible ways $\mathcal{A}_2$ could learn $F(r^*)$:

1. $\mathcal{A}_2$ gets lucky in one of his $\mathrm{poly}(n)$ queries to Enc_k and happens to sample $(r^*, F(r^*))$;
2. Or, $\mathcal{A}$ is somehow able to use what he learned while he had access to Dec_k, and thus F, to learn $F(r^*)$, meaning that the $\mathrm{poly}(n)$-sized quantum memory $\mathcal{A}_1$ sends to $\mathcal{A}_2$, that can depend on queries to F, but which cannot depend on r^*, allows $\mathcal{A}_2$ to learn $F(r^*)$.

The first possibility is exponentially unlikely, since there are 2^n possibilities for r^*. As we will see shortly, the second possibility would imply a very strong quantum random access code. It would essentially allow $\mathcal{A}_1$ to interact with F, which contains 2^n values, and make a state, which must necessarily be of polynomial size, such that $\mathcal{A}_2$ can use that state to recover $F(r^*)$ for any of the 2^n possible values of r^*, with high probability. We now formalize this intuition. To clarify notation, we will use boldface to denote the shared randomness bitstrings.

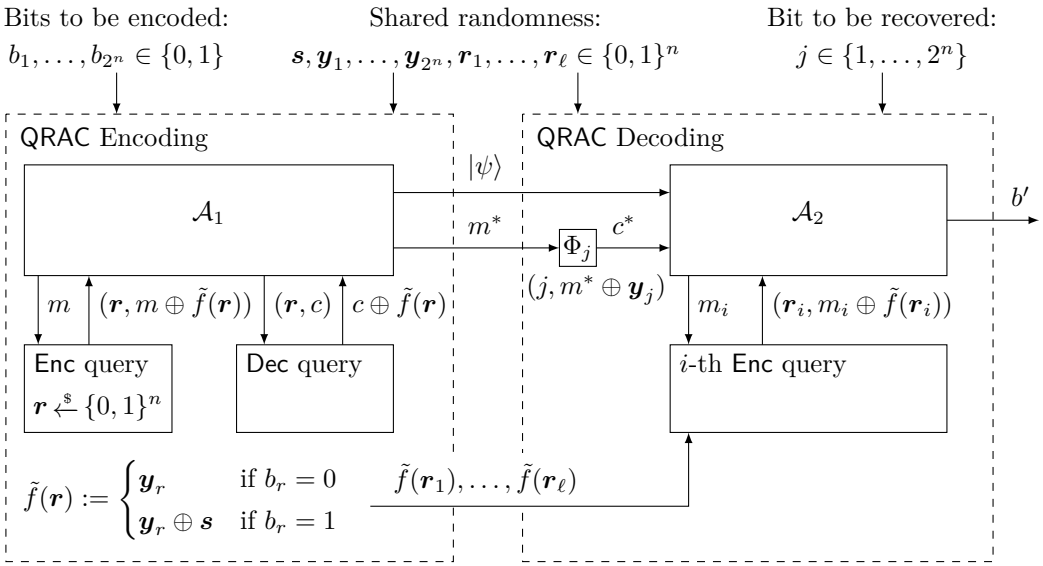


Figure 2 Quantum random access code construction for the PRF scheme.

Construction of a quantum random access code. Let $\mathcal{A}$ be a QPT adversary with winning probability p. Let $\ell = \mathrm{poly}(n)$ be an upper bound on the number of queries made by $\mathcal{A}_2$. Recall that a random access code consists of an encoding procedure that takes (in this case) 2^n bits $b_1, \ldots, b_{2^n}$, and outputs a state ϱ of dimension (in this case) $2^{\mathrm{poly}(n)}$, such that a decoding procedure, given ϱ and an index $j \in \{1, \ldots, 2^n\}$ outputs b_j with some success probability. We define a quantum random access code as follows (see also Figure 2).

Encoding. Let $b_1, \ldots, b_{2^n} \in \{0,1\}$ be the string to be encoded. Let $\boldsymbol{s}, \boldsymbol{y}_1, \ldots, \boldsymbol{y}_{2^n} \in \{0,1\}^n$ be the first $n(1+2^n)$ bits of the shared randomness, and let $\boldsymbol{r}_1, \ldots, \boldsymbol{r}_\ell \in \{0,1\}^n$ be the next ℓn bits. Define $\tilde{f} : \{0,1\}^n \to \{0,1\}^n$ as follows. For $\boldsymbol{r} \in \{0,1\}^n$, we slightly abuse notation by letting r denote the corresponding integer value between 1 and 2^n. Define $\tilde{f}(\boldsymbol{r}) = \boldsymbol{y}_r \oplus b_r \boldsymbol{s}$. Run $\mathcal{A}_1$, answering encryption and decryption queries using $\tilde{f}$ in place of F. Let m^* and $|\psi\rangle$ be the outputs of $\mathcal{A}_1$ (see Figure 1). Output $\varrho = (|\psi\rangle, m^*, \tilde{f}(\boldsymbol{r}_1), \ldots, \tilde{f}(\boldsymbol{r}_\ell))$.

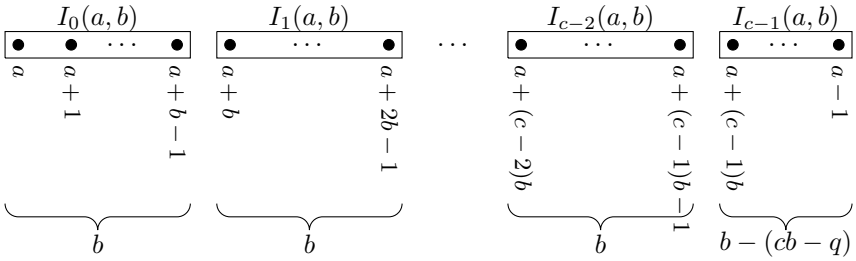


Figure 3 Dividing $\mathbb{Z}_q$ into $c = \lceil q/b \rceil$ blocks, starting from a. The first $c-1$ blocks, labelled $I_0(a,b), \ldots, I_{c-2}(a,b)$, have size b and the last, labelled $I_{c-1}(a,b)$, contains the remaining $b-(cb-q) \leq b$ elements of $\mathbb{Z}_q$.

Decoding. Let $j \in \{1, \ldots, 2^n\}$ be the index of the bit to be decoded (so given ϱ as above, the goal is to recover b_j). Decoding will make use of the values $\boldsymbol{s}, \boldsymbol{y}_1, \ldots, \boldsymbol{y}_{2^n}, \boldsymbol{r}_1, \ldots, \boldsymbol{r}_\ell$ given by the shared randomness. Upon receiving a query $j \in \{1, \ldots, 2^n\}$, run $\mathcal{A}_2$ with inputs $|\psi\rangle$ and $(j, m^* \oplus \boldsymbol{y}_j)$. On $\mathcal{A}_2$'s i-th encryption oracle call, use randomness $\boldsymbol{r}_i$, so that if the input to the oracle is $|m, c\rangle$, the state returned is $|m, c \oplus (\boldsymbol{r}_i, m \oplus \tilde{f}(\boldsymbol{r}_i))\rangle$ (note that $\tilde{f}(\boldsymbol{r}_i)$ is given as part of ϱ). Return the bit b' output by $\mathcal{A}_2$.

Average bias of the code. We claim that the average probability of decoding correctly, taken over all choices of $b_1, \ldots, b_{2^n} \in \{0,1\}$ and $j \in \{1, \ldots, 2^n\}$, is exactly p, the success probability of $\mathcal{A}$. To see this, first note that from $\mathcal{A}$'s perspective, this is exactly Game 1: the function $\tilde{f}$ is a uniformly random function, and the queries are responded to just as in Game 1. Further, note that if $b_j = 0$, then $m^* \oplus \boldsymbol{y}_j = m^* \oplus \tilde{f}(j)$, so the correct guess for $\mathcal{A}_2$ would be 0, and if $b_j = 1$, then $m^* \oplus \boldsymbol{y}_j = m^* \oplus \tilde{f}(j) \oplus \boldsymbol{s} = \boldsymbol{x} \oplus \tilde{f}(j)$ for the uniformly random string $\boldsymbol{x} = m^* \oplus \boldsymbol{s}$, so the correct guess for $\mathcal{A}_2$ would be 1.

Therefore, the average bias of the code is $p - 1/2$. We also observe that ϱ has dimension at most $2^{\mathrm{poly}(n)}$, since $|\psi\rangle$ must be a poly(n)-qubit state ($\mathcal{A}_1$ only runs for poly(n) time), and ℓ, the number of queries made by $\mathcal{A}_2$ must be poly(n), since $\mathcal{A}_2$ only runs for poly(n) time. As this code encodes 2^n bits into a state of dimension $2^{\mathrm{poly}(n)}$, by Lemma 6 (proven in Appendix A), the bias is $O(2^{-n/2}\,\mathrm{poly}(n)) = \mathrm{negl}(n)$, so $p \leq \frac{1}{2} + \mathrm{negl}(n)$. ◀

5 Quantum algorithm for linear rounding functions

In this section, we analyze the performance of the Bernstein-Vazirani algorithm [3] with a modified version of the oracle. While the original oracle computes the inner product modulo q, our version only gives partial information about it by rounding its value to one of $\lceil q/b \rceil$ blocks of size b, for some $b \in \{1, \ldots, q-1\}$ (if b does not divide q, one of the blocks will have size $< b$).

▶ **Definition 15.** *Let $n \geq 1$ be an integer and $q \geq 2$ be an integer modulus. Let $a \in \mathbb{Z}_q$, $b \in \mathbb{Z}_q \setminus \{0\}$ and $c := \lceil q/b \rceil$. We partition $\mathbb{Z}_q$ into c disjoint blocks (most of them of size b) starting from a as follows (see Figure 3):*

$$I_v(a,b) := \begin{cases} \{a + vb, \ldots, a + vb + b - 1\} & \text{if } v \in \{0, \ldots, c-2\}, \\ \{a + vb, \ldots, a + q - 1\} & \text{if } v = c-1. \end{cases}$$

Based on this partition, we define a family $\mathsf{LRF}_{\boldsymbol{k},a,b} : \mathbb{Z}_q^n \longrightarrow \mathbb{Z}_c$ *of keyed linear rounding functions, with key* $\boldsymbol{k} \in \mathbb{Z}_q^n$*, as follows:*

$$\mathsf{LRF}_{\boldsymbol{k},a,b}(\boldsymbol{x}) := v \text{ if } \langle \boldsymbol{x}, \boldsymbol{k} \rangle \in I_v(a,b).$$

Algorithm 1: Bernstein-Vazirani for linear rounding functions.

Parameters: n, q, $b \in \{1, \ldots, q-1\}$, $c = \lceil q/b \rceil$.

Input: Quantum oracle $U_{\mathsf{LRF}} : |\boldsymbol{x}\rangle|z\rangle \mapsto |\boldsymbol{x}\rangle|z + \mathsf{LRF}_{\boldsymbol{k},a,b}(\boldsymbol{x}) \pmod c\rangle$ where $\boldsymbol{x} \in \mathbb{Z}_q^n$, $z \in \mathbb{Z}_c$ and $\mathsf{LRF}_{\boldsymbol{k},a,b}$ is the rounded inner product function for some unknown $\boldsymbol{k} \in \mathbb{Z}_q^n$ and $a \in \mathbb{Z}_q$.

Output: String $\tilde{\boldsymbol{k}} \in \mathbb{Z}_q^n$ such that $\tilde{\boldsymbol{k}} = \boldsymbol{k}$ with high probability.

1. Prepare the uniform superposition and append $\frac{1}{\sqrt{c}} \sum_{z=0}^{c-1} \omega_c^z |z\rangle$ where $\omega_c = e^{2\pi i/c}$: $\frac{1}{\sqrt{q^n}} \sum_{\boldsymbol{x} \in \mathbb{Z}_q^n} |\boldsymbol{x}\rangle \otimes \frac{1}{\sqrt{c}} \sum_{z=0}^{c-1} \omega_c^z |z\rangle$.
2. Query oracle U_{LRF} for $\mathsf{LRF}_{\boldsymbol{k},a,b}$; obtain $\frac{1}{\sqrt{q^n}} \sum_{\boldsymbol{x} \in \mathbb{Z}_q^n} \omega_c^{-\mathsf{LRF}_{\boldsymbol{k},a,b}(\boldsymbol{x})} |\boldsymbol{x}\rangle \otimes \frac{1}{\sqrt{c}} \sum_{z=0}^{c-1} \omega_c^z |z\rangle$.
3. Discard the last register and apply the quantum Fourier transform $\mathsf{QFT}_{\mathbb{Z}_q}^{\otimes n}$.
4. Measure in the computational basis and output the outcome $\tilde{\boldsymbol{k}}$.

The following theorem shows that the modulo-q variant of the Bernstein-Vazirani algorithm (Algorithm 1) can recover $\boldsymbol{k}$ with constant probability of success by using only a single quantum query to $\mathsf{LRF}_{\boldsymbol{k},a,b}$. The proof is a fairly straightforward computation through the steps of the algorithm, and can be found in Appendix C.

▶ **Theorem 16.** *Let* U_{LRF} *be the quantum oracle for the linear rounding function* $\mathsf{LRF}_{\boldsymbol{k},a,b}$ *with modulus* $q \geq 2$*, block size* $b \in \{1, \ldots, q-1\}$*, and an unknown* $a \in \{0, \ldots, q-1\}$*, and unknown key* $\boldsymbol{k} \in \mathbb{Z}_q^n$ *such that* $\boldsymbol{k}$ *has at least one entry that is a unit modulo* q*. Let* $c = \lceil q/b \rceil$ *and* $d = cb - q$*. By making one query to the oracle* U_{LRF}*, Algorithm 1 recovers the key* $\boldsymbol{k}$ *with probability at least* $4/\pi^2 - O(d/q)$*.*

6 Key recovery against LWE

In this section, we consider various LWE-based encryption schemes and show using Theorem 16 that the decryption key can be efficiently recovered using a single quantum decryption query. In the full version of this work, we also show that a single quantum *encryption* query can be used to recover the secret key in a symmetric-key version of LWE, as long as the querying algorithm also has control over part of the randomness used in the encryption procedure.

6.1 Key recovery via one decryption query in symmetric-key LWE

Recall the following standard construction of an IND-CPA symmetric-key encryption scheme based on the LWE assumption [23].

▶ **Construction 17** (LWE-SKE [23])**.** *Let* $n \geq 1$ *be an integer, let* $q \geq 2$ *be an integer modulus and let* χ *be a discrete and symmetric error distribution. Then, the symmetric-key encryption scheme* $\mathsf{LWE\text{-}SKE}(n, q, \chi) = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ *is defined as follows:*

1. KeyGen: *output* $\boldsymbol{k} \xleftarrow{\$} \mathbb{Z}_q^n$;
2. $\mathsf{Enc}_{\boldsymbol{k}}$: *to encrypt* $b \in \{0,1\}$, *sample* $\boldsymbol{a} \xleftarrow{\$} \mathbb{Z}_q^n$, $e \xleftarrow{\chi} \mathbb{Z}_q$ *and output* $(\boldsymbol{a}, \langle \boldsymbol{a}, \boldsymbol{k} \rangle + b\lfloor \frac{q}{2} \rfloor + e)$;
3. $\mathsf{Dec}_{\boldsymbol{k}}$: *to decrypt* $(\boldsymbol{a}, c)$, *output* 0 *if* $|c - \langle \boldsymbol{a}, \boldsymbol{k} \rangle| \leq \lfloor \frac{q}{4} \rfloor$, *else output* 1.

As a corollary of Theorem 16, an adversary that is granted a single quantum decryption query can recover the key with probability at least $4/\pi^2 - o(1)$:

▶ **Corollary 18.** *There is a quantum algorithm that makes a single quantum query to* $\mathsf{LWE\text{-}SKE.Dec}_{\boldsymbol{k}}$ *and recovers the entire key* $\boldsymbol{k}$ *with probability at least* $4/\pi^2 - o(1)$.

Proof. $\mathsf{LWE\text{-}SKE.Dec}_{\boldsymbol{k}}$ coincides with a linear rounding function $\mathsf{LRF}_{\boldsymbol{k}',a,b}$ for a key $\boldsymbol{k}' = (-\boldsymbol{k}, 1) \in \mathbb{Z}_q^{n+1}$, which has a unit in its last entry. In particular, $b = \lceil q/2 \rceil$, and if $q = 3 \pmod 4$, $a = \lceil q/4 \rceil$, and otherwise, $a = -\lfloor q/4 \rfloor$. Thus, by Theorem 16, Algorithm 1 makes one quantum query to $\mathsf{LRF}_{\boldsymbol{k}',a,b}$, which can be implemented using one quantum query to $\mathsf{LWE\text{-}SKE.Dec}_{\boldsymbol{k}}$, and recovers $\boldsymbol{k}'$, and thus $\boldsymbol{k}$, with probability $4/\pi^2 - O(d/q)$, where $d = \lceil q/b \rceil b - q \leq 1$. ◀

Note that the key in this scheme consists of $n \log q$ uniformly random bits, and that a classical decryption query yields at most a single bit of output. It follows that any algorithm making t *classical* queries to the decryption oracle recovers the entire key with probability at most $2^{t - n \log q}$. A straightforward key-recovery algorithm does in fact achieve this.

6.2 Key recovery via one decryption query in public-key LWE

The key-recovery attack described in Corollary 18 required nothing more than the fact that the decryption procedure of LWE-SKE is just a linear rounding function whose key contains the decryption key. As a result, the attack is naturally applicable to other variants of LWE. In this section, we consider two public-key variants. The first is the standard construction of IND-CPA public-key encryption based on the LWE assumption, as introduced by Regev [23]. The second is the IND-CPA-secure public-key encryption scheme FrodoPKE [1], which is based on a construction of Lindner and Peikert [18]. In both cases, we demonstrate a dramatic speedup in key recovery using quantum decryption queries.

We emphasize once again that key recovery against these schemes was already possible classically using a linear number of decryption queries. Our results should thus not be interpreted as a weakness of these cryptosystems in their stated security setting (i.e., IND-CPA). The proper interpretation is that, if these cryptosystems are exposed to chosen-ciphertext attacks, then quantum attacks can be even more devastating than classical ones.

6.2.1 Regev's public-key scheme

The standard construction of an IND-CPA public-key encryption scheme based on LWE is the following.

▶ **Construction 19** (LWE-PKE [23])**.** *Let* $m \geq n \geq 1$ *be integers, let* $q \geq 2$ *be an integer modulus, and let* χ *be a discrete error distribution over* $\mathbb{Z}_q$. *Then, the public-key encryption scheme* $\mathsf{LWE\text{-}PKE}(n, q, \chi) = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ *is defined as follows:*

1. KeyGen: *output a secret key* $\boldsymbol{sk} = \boldsymbol{k} \xleftarrow{\$} \mathbb{Z}_q^n$ *and a public key* $\boldsymbol{pk} = (\boldsymbol{A}, \boldsymbol{Ak} + \boldsymbol{e}) \in \mathbb{Z}_q^{m \times (n+1)}$, *where* $\boldsymbol{A} \xleftarrow{\$} \mathbb{Z}_q^{m \times n}$, $\boldsymbol{e} \xleftarrow{\chi} \mathbb{Z}_q^m$, *and all arithmetic is done modulo* q.
2. Enc: *to encrypt* $b \in \{0,1\}$, *pick a random* $\boldsymbol{v} \in \{0,1\}^m$ *with Hamming weight roughly* $m/2$ *and output* $(\boldsymbol{v}^\mathsf{T}\boldsymbol{A}, \boldsymbol{v}^\mathsf{T}(\boldsymbol{Ak} + \boldsymbol{e}) + b\lfloor \frac{q}{2} \rfloor) \in \mathbb{Z}_q^{n+1}$, *where* $\boldsymbol{v}^\mathsf{T}$ *denotes the transpose of* $\boldsymbol{v}$.
3. Dec: *to decrypt* $(\boldsymbol{a}, c)$, *output* 0 *if* $|c - \langle \boldsymbol{a}, \boldsymbol{sk} \rangle| \leq \lfloor \frac{q}{4} \rfloor$, *else output* 1.

Although the encryption is now done in a public-key manner, all that matters for our purposes is the decryption procedure, which is identical to the symmetric-key case, LWE-SKE. We thus have the following corollary, whose proof is identical to that of Corollary 18:

▶ **Corollary 20.** *There is a quantum algorithm that makes a single quantum query to* $\mathsf{LWE\text{-}PKE.Dec}_{\boldsymbol{sk}}$ *and recovers the entire key* $\boldsymbol{sk}$ *with probability at least* $4/\pi^2 - o(1)$.

6.2.2 Frodo public-key scheme

Next, we consider the IND-CPA-secure public-key encryption scheme FrodoPKE, which is based on a construction by Lindner and Peikert [18]. Compared to LWE-PKE, this scheme significantly reduces the key-size and achieves better security estimates than the initial proposal by Regev [23]. For a detailed discussion of FrodoPKE, we refer to [1]. We present the entire scheme for completeness, but the important part for our purposes is the decryption procedure.

▶ **Construction 21** (FrodoPKE [1])**.** *Let* $n, \bar{m}, \bar{n}$ *be integer parameters,* $q \geq 2$ *an integer power of 2. Let* B *denote the number of bits used for encoding, and let* χ *be a discrete symmetric error distribution. The public-key encryption scheme* $\mathsf{FrodoPKE} = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ *is defined:*

1. KeyGen: *generate a matrix* $\boldsymbol{A} \xleftarrow{\$} \mathbb{Z}_q^{n\times n}$ *and matrices* $\boldsymbol{S}, \boldsymbol{E} \xleftarrow{\chi} \mathbb{Z}_q^{n\times\bar{n}}$*; compute* $\boldsymbol{B} = \boldsymbol{AS} + \boldsymbol{E} \in \mathbb{Z}_q^{n\times\bar{n}}$*; output the key-pair* $(\boldsymbol{pk}, \boldsymbol{sk})$ *with public key* $\boldsymbol{pk} = (\boldsymbol{A}, \boldsymbol{B})$ *and secret key* $\boldsymbol{sk} = \boldsymbol{S}$.
2. Enc: *to encrypt* $\boldsymbol{m} \in \{0,1\}^{B\cdot\bar{m}\cdot\bar{n}}$ *(encoded as a matrix* $\boldsymbol{M} \in \mathbb{Z}_q^{\bar{m}\times\bar{n}}$ *with each entry having 0s in all but the* B *most significant bits) with public key* $\boldsymbol{pk}$*, sample error matrices* $\boldsymbol{S}', \boldsymbol{E}' \xleftarrow{\chi} \mathbb{Z}_q^{\bar{m}\times n}$ *and* $\boldsymbol{E}'' \xleftarrow{\chi} \mathbb{Z}_q^{\bar{m}\times\bar{n}}$*; compute* $\boldsymbol{C}_1 = \boldsymbol{S}'\boldsymbol{A} + \boldsymbol{E}' \in \mathbb{Z}_q^{\bar{m}\times n}$ *and* $\boldsymbol{C}_2 = \boldsymbol{M} + \boldsymbol{S}'\boldsymbol{B} + \boldsymbol{E}'' \in \mathbb{Z}_q^{\bar{m}\times\bar{n}}$*; output the ciphertext* $(\boldsymbol{C}_1, \boldsymbol{C}_2)$.
3. Dec: *to decrypt* $(\boldsymbol{C}_1, \boldsymbol{C}_2) \in \mathbb{Z}_q^{\bar{m}\times n} \times \mathbb{Z}_q^{\bar{m}\times\bar{n}}$ *with secret-key* $\boldsymbol{sk} = \boldsymbol{S}$*, compute* $\boldsymbol{M} = \boldsymbol{C}_2 - \boldsymbol{C}_1\boldsymbol{S} \in \mathbb{Z}_q^{\bar{m}\times\bar{n}}$*. For each* $(i,j) \in [\bar{m}] \times [\bar{n}]$*, output the first* B *bits of* $M_{i,j}$.

We now show how to recover $\bar{m}$ of the $\bar{n}$ columns of the secret key $\boldsymbol{S}$ using a single quantum query to $\mathsf{FrodoPKE.Dec}_{\boldsymbol{S}}$. If $\bar{m} = \bar{n}$, as in sample parameters given in [1], then this algorithm recovers $\boldsymbol{S}$ completely.

▶ **Theorem 22.** *There exists a quantum algorithm that makes one quantum query to* $\mathsf{FrodoPKE.Dec}_{\boldsymbol{S}}$ *and recovers any choice of* $\bar{m}$ *of the* $\bar{n}$ *columns of* $\boldsymbol{S}$*. For each of the chosen columns, if that column has at least one odd entry, then the algorithm succeeds in recovering the column with probability at least* $4/\pi^2$.

We give a formal proof of Theorem 22 in Appendix D, but here we briefly sketch the proof. Let $\boldsymbol{s}^1, \ldots, \boldsymbol{s}^{\bar{n}}$ be the columns of $\boldsymbol{S}$. Let U denote the map:

$$U : |\boldsymbol{c}\rangle|z_1\rangle \ldots |z_{\bar{n}}\rangle \mapsto |\boldsymbol{c}\rangle|z_1 + \mathsf{LRF}_{\boldsymbol{s}^1,0,q/2^B}(\boldsymbol{c})\rangle \ldots |z_{\bar{n}} + \mathsf{LRF}_{\boldsymbol{s}^{\bar{n}},0,q/2^B}(\boldsymbol{c})\rangle,$$

for any $\boldsymbol{c} \in \mathbb{Z}_q^n$ and $z_1, \ldots, z_{\bar{n}} \in \mathbb{Z}_{2^B}$. By a straightforward calculation, one can show that a single call to $\mathsf{FrodoKEM.Dec}_{\boldsymbol{S}}$, with $\boldsymbol{C}_2$ set of $0^{\bar{m}\times\bar{n}}$, can be used to implement $U^{\otimes\bar{m}}$. Then we show that one call to U can be used to recover any choice of the columns of $\boldsymbol{S}$ with probability $4/\pi^2$, as long as it has at least one entry that is odd. To show this, we show that U can be used to implement a phase query to $\mathsf{LRF}_{\boldsymbol{s}^j,0,q/2^B}$, by simply applying U to a state with $|\varphi\rangle = 2^{-B/2}\sum_{z=0}^{2^B-1}|z\rangle$ in each of $\bar{n}$ registers except the j-th one, and $\frac{1}{\sqrt{2^B}}\sum_{z=0}^{2^B-1}\omega_{2^B}^z|z\rangle$ in the j-th register. This ensures that the phase corresponding to $\mathsf{LRF}_{\boldsymbol{s}^j,0,q/2^B}$ is kicked back, but all other phases, corresponding to $\mathsf{LRF}_{\boldsymbol{s}^{j'},0,q/2^B}$ for $j' \neq j$ are not. For details, see Appendix D.

6.3 Key recovery via one decryption query in public-key Ring-LWE

Next, we analyze key-recovery with a single quantum decryption query against Ring-LWE encryption. Unlike the plain LWE-based encryption schemes we considered in the previous sections, Ring-LWE encryption uses noisy samples over a polynomial ring. In the following, we consider the basic, bit-by-bit Ring-LWE public-key encryption scheme introduced in [19, 20]. It is based on the rings $\mathcal{R} = \mathbb{Z}[x]/\langle x^n + 1\rangle$ and $\mathcal{R}_q := \mathcal{R}/q\mathcal{R} = \mathbb{Z}_q[x]/\langle x^n + 1\rangle$ for some power-of-two integer n and poly(n)-bounded prime modulus q. The details of the error distribution χ below will not be relevant to our results.

▶ **Construction 23** (Ring-LWE-PKE [19, 20])**.** *Let $n \geq 1$ be an integer, let $q \geq 2$ be an integer modulus, and let χ be an error distribution over $\mathcal{R}$. The public-key encryption scheme* Ring-LWE-PKE $=$ (KeyGen, Enc, Dec) *is defined as follows:*

1. KeyGen*: sample $a \xleftarrow{\$} \mathcal{R}_q$ and $e, s \xleftarrow{\chi} \mathcal{R}$; output $sk = s$ and $pk = (a, c = a \cdot s + e \pmod q) \in \mathcal{R}_q^2$.*
2. Enc*: to encrypt $b \in \{0, 1\}$, sample $r, e_1, e_2 \xleftarrow{\chi} \mathcal{R}$ and output a ciphertext pair $(u, v) \in \mathcal{R}_q^2$, where $u = a \cdot r + e_1 \pmod q$ and $v = c \cdot r + e_2 + b\lfloor q/2 \rfloor \pmod q$.*
3. Dec*: to decrypt (u, v), compute $v - u \cdot s = (r \cdot e - s \cdot e_1 + e_2) + b\lfloor q/2 \rfloor \pmod q \in \mathcal{R}_q$; output 0 if the constant term of the polynomial is closer to 0 than $\lfloor q/2 \rfloor$, else output 1.*

We note that our choice of placing single-bit encryption in the constant term of the polynomial is somewhat arbitrary. Indeed, it is straightforward to extend our results to encryption with respect to other monomials.

In the full version of the article, we show the following corollary to Theorem 16.

▶ **Corollary 24.** *There is a quantum algorithm that makes one quantum query to* Ring-LWE-PKE.Dec$_s$ *and recovers the entire key s with probability at least $4/\pi^2 - o(1)$.*

References

1 Erdem Alkim, Joppe W. Bos, Léo Ducas, Patrick Longa, Ilya Mironov, Michael Naehrig, Valeria Nikolaenko, Chris Peikert, Ananth Raghunathan, Douglas Stebila, Karen Easterbrook, and Brian LaMacchia. *FrodoKEM – Learning With Errors Key Encapsulation*, 2017. URL: https://frodokem.org/files/FrodoKEM-specification-20171130.pdf.

2 Andris Ambainis, Debbie Leung, Laura Mancinska, and Maris Ozols. Quantum random access codes with shared randomness, 2008. arXiv:0810.2937.

3 Ethan Bernstein and Umesh Vazirani. Quantum complexity theory. *SIAM Journal on Computing*, 26(5):1411–1473, 1997. doi:10.1137/S0097539796300921.

4 Daniel Bleichenbacher. Chosen ciphertext attacks against protocols based on the RSA encryption standard PKCS #1. In Hugo Krawczyk, editor, *Advances in Cryptology – CRYPTO '98*, pages 1–12. Springer, 1998. doi:10.1007/BFb0055716.

5 Dan Boneh and Mark Zhandry. Quantum-secure message authentication codes. In Thomas Johansson and Phong Q. Nguyen, editors, *Advances in Cryptology – EUROCRYPT 2013*, pages 592–608. Springer, 2013. doi:10.1007/978-3-642-38348-9_35.

6 Dan Boneh and Mark Zhandry. Secure signatures and chosen ciphertext security in a quantum computing world. In Ran Canetti and Juan A. Garay, editors, *Advances in Cryptology – CRYPTO 2013*, pages 361–379. Springer, 2013. doi:10.1007/978-3-642-40084-1_21.

7 Anne Broadbent and Stacey Jeffery. Quantum homomorphic encryption for circuits of low T-gate complexity. In Rosario Gennaro and Matthew Robshaw, editors, *Advances in Cryptology – CRYPTO 2015*, pages 609–629. Springer, 2015. doi:10.1007/978-3-662-48000-7_30.

8 Lily Chen, Stephen Jordan, Yi-Kai Liu, Dustin Moody, Rene Peralta, Ray Perlner, and Daniel Smith-Tone. Report on post-quantum cryptography. Technical report, National Institute of Standards and Technology, 2016. doi:10.6028/NIST.IR.8105.

9 Eiichiro Fujisaki and Tatsuaki Okamoto. Secure integration of asymmetric and symmetric encryption schemes. In *Advances in Cryptography – CRYPTO 1999*, pages 537–554, 1999.

10 Tommaso Gagliardoni, Andreas Hülsing, and Christian Schaffner. Semantic security and indistinguishability in the quantum world. In Matthew Robshaw and Jonathan Katz, editors, *Advances in Cryptology – CRYPTO 2016*, pages 60–89. Springer, 2016. doi:10.1007/978-3-662-53015-3_3.

11 Oded Goldreich. *Foundations of Cryptography: Volume 2, Basic Applications*. Cambridge University Press, Cambridge, UK, 2009.

12 Oded Goldreich, Shafi Goldwasser, and Silvio Micali. How to construct random functions. *Journal of the ACM*, 33(4):792–807, 1986. doi:10.1145/6490.6503.

13 Oded Goldreich and Leonid A. Levin. A hard-core predicate for all one-way functions. In *Proceedings of the Twenty-first Annual ACM Symposium on Theory of Computing*, STOC '89, pages 25–32, New York, NY, USA, 1989. ACM. doi:10.1145/73007.73010.

14 Alex B. Grilo, Iordanis Kerenidis, and Timo Zijlstra. Learning with errors is easy with quantum samples, 2017. arXiv:1702.08255.

15 Marc Kaplan, Gaëtan Leurent, Anthony Leverrier, and María Naya-Plasencia. Breaking symmetric cryptosystems using quantum period finding. In Matthew Robshaw and Jonathan Katz, editors, *Advances in Cryptology – CRYPTO 2016*, pages 207–237. Springer, 2016. doi:10.1007/978-3-662-53008-5_8.

16 Hidenori Kuwakado and Masakatu Morii. Quantum distinguisher between the 3-round Feistel cipher and the random permutation. In *2010 IEEE International Symposium on Information Theory*, pages 2682–2685. IEEE, 2010. doi:10.1109/ISIT.2010.5513654.

17 Hidenori Kuwakado and Masakatu Morii. Security on the quantum-type Even-Mansour cipher. In *2012 International Symposium on Information Theory and its Applications*, pages 312–316. IEEE, 2012. URL: https://ieeexplore.ieee.org/document/6400943/.

18 Richard Lindner and Chris Peikert. Better key sizes (and attacks) for LWE-based encryption. In Aggelos Kiayias, editor, *Topics in Cryptology – CT-RSA 2011*, pages 319–339, Berlin, Heidelberg, 2011. Springer. doi:10.1007/978-3-642-19074-2_21.

19 Vadim Lyubashevsky, Chris Peikert, and Oded Regev. A toolkit for ring-LWE cryptography. In Thomas Johansson and Phong Q. Nguyen, editors, *Advances in Cryptology – EUROCRYPT 2013*, pages 35–54. Springer, 2013. doi:10.1007/978-3-642-38348-9_3.

20 Vadim Lyubashevsky, Chris Peikert, and Oded Regev. On ideal lattices and learning with errors over rings. *Journal of the ACM*, 60(6):43:1–43:35, November 2013. doi:10.1145/2535925.

21 Ashwin Nayak. Optimal lower bounds for quantum automata and random access codes. In *40th Annual Symposium on Foundations of Computer Science*, pages 369–376, 1999. doi:10.1109/SFFCS.1999.814608.

22 NIST. Post-Quantum Cryptography, 2017. URL: https://csrc.nist.gov/projects/post-quantum-cryptography.

23 Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. *Journal of the ACM*, 56(6):34:1–34:40, 2005. doi:10.1145/1568318.1568324.

24 Thomas Santoli and Christian Schaffner. Using Simon's algorithm to attack symmetric-key cryptographic primitives. *Quantum Information & Computation*, 17(1&2):65–78, 2017. doi:10.26421/QIC17.1-2.

25 Peter W. Shor. Algorithms for quantum computation: discrete logarithms and factoring. In *Proceedings of the 35th Annual Symposium on Foundations of Computer Science*, pages 124–134. IEEE, 1994. doi:10.1109/SFCS.1994.365700.

26 Daniel R. Simon. On the power of quantum computation. *SIAM Journal on Computing*, 26(5):1474–1483, 1997. doi:10.1137/S0097539796298637.

27 Nicole Tomczak-Jaegermann. The moduli of smoothness and convexity and the Rademacher averages of the trace classes S_p $(1 \leq p < \infty)$. *Studia Mathematica*, 50(2):163–182, 1974. URL: http://eudml.org/doc/217886.

28 Mark Zhandry. How to construct quantum random functions. In *2012 IEEE 53rd Annual Symposium on Foundations of Computer Science*, pages 679–687. IEEE, 2012. doi:10.1109/FOCS.2012.37.

29 Mark Zhandry. A note on quantum-secure PRPs, 2016. arXiv:1611.05564.

A Bound for quantum random access codes

A variation of quantum random access codes allows Alice and Bob to use *shared randomness* in their encoding and decoding operations [2] (note that shared randomness *per se* does not allow them to communicate). We are interested in bounding the average bias $\epsilon = p_{\text{win}} - 1/2$ of a quantum random access code with shared randomness, where p_{win} is the winning probability averaged over $x \xleftarrow{\$} \{0,1\}^N$ and $i \xleftarrow{\$} \{1,\dots,N\}$.

▶ **Lemma 25.** *The average bias of a quantum random access code with shared randomness that encodes N bits into a d-dimensional quantum state is $O(\sqrt{N^{-1}\log d})$. In particular, if $N = 2^n$ and $d = 2^{\text{poly}(n)}$ the bias is $O(2^{-n/2}\,\text{poly}(n))$.*

Proof. A quantum random access code with shared randomness that encodes N bits into a d-dimensional quantum state is specified by the following:

- a shared random variable λ,
- for each $x \in \{0,1\}^N$, a d-dimensional quantum state ϱ_x^λ encoding x,
- for each $i \in \{0,\dots,N\}$, an observable M_i^λ for recovering the i-th bit.

Formally, ϱ_x^λ and M_i^λ are $d \times d$ Hermitian matrices such that $\varrho_x^\lambda \geq 0$, $\text{Tr}\varrho_x^\lambda = 1$, and $\|M_i^\lambda\| \leq 1$ where $\|M_i^\lambda\|$ denotes the operator norm of M_i^λ. Note that both ϱ_x^λ and M_i^λ depend on the shared random variable λ, meaning that Alice and Bob can coordinate their strategies.

The bias of correctly guessing x_i, for a given x and i, is $(-1)^{x_i}\text{Tr}(\varrho_x^\lambda M_i^\lambda)/2$. If the average bias of the code is ϵ then $\mathbb{E}_\lambda \mathbb{E}_{x,i}(-1)^{x_i}\text{Tr}(\varrho_x^\lambda M_i^\lambda) \geq 2\epsilon$. We can rearrange this expression and upper bound each term using its operator norm, and then apply the noncommutative Khintchine inequality [27]:

$$\begin{aligned}\mathbb{E}_\lambda \mathbb{E}_x \frac{1}{N}\text{Tr}\Big(\varrho_x^\lambda \sum_{i=1}^N (-1)^{x_i} M_i^\lambda\Big) &\leq \mathbb{E}_\lambda \mathbb{E}_x \frac{1}{N}\|\sum_{i=1}^N (-1)^{x_i} M_i^\lambda\| \\ &\leq \mathbb{E}_\lambda \frac{1}{N} c\sqrt{N\log d} = c\sqrt{\frac{\log d}{N}},\end{aligned}$$

for some constant c. In other words, $\epsilon \leq \frac{c}{2}\sqrt{\frac{\log d}{N}}$. In the particular case we are interested in, $d = 2^{\text{poly}(n)}$ and $N = 2^n$ so $\epsilon \leq \frac{c}{2}\sqrt{\frac{\text{poly}(n)}{2^n}}$, completing the proof. ◀

B Equivalence of QCCA1 models

Recall that the IND-QCCA1 notion is based on the security game IndGame defined in Definition 9. In the alternative security game IndGame′ (see Definition 10), the adversary provides only one plaintext m and must decide if the challenge is an encryption of m or an encryption of a random string. In this section, we prove the following:

▶ **Proposition 26.** *An encryption scheme Π is* IND-QCCA1 *if and only if for every* QPT *$\mathcal{A}$,*

$$\Pr[\mathcal{A} \text{ wins } \mathsf{IndGame}'(\Pi,\mathcal{A},n)] \leq 1/2 + \text{negl}(n)\,.$$

Proof. Fix a scheme Π. For one direction, suppose Π is IND-QCCA1 and let $\mathcal{A}$ be an adversary against IndGame′. Define an adversary $\mathcal{A}_0$ against IndGame as follows: (i.) run $\mathcal{A}$ until it outputs a challenge plaintext m, (ii.) sample random r and output (m, r), (iii.) run the rest of $\mathcal{A}$ and output what it outputs. The output distribution of $\mathsf{IndGame}'(\Pi, \mathcal{A}, n)$ is then identical to $\mathsf{IndGame}(\Pi, \mathcal{A}_0, n)$, which in turn must be negligibly close to uniform by IND-QCCA1 security of Π.

For the other direction, suppose no adversary can win IndGame′ with probability better than $1/2$, and let $\mathcal{B}$ be an adversary against IndGame. Now, define two adversaries $\mathcal{B}_0$ and $\mathcal{B}_1$ against IndGame′ as follows. The adversary $\mathcal{B}_c$ does: (i.) run $\mathcal{B}$ until it outputs a challenge (m_0, m_1), (ii.) output m_c, (iii.) run the rest of $\mathcal{B}$ and output what it outputs. Note that the pre-challenge algorithm is identical for $\mathcal{B}$, $\mathcal{B}_0$, and $\mathcal{B}_1$; define random variables M_0, M_1 and R given by the two challenges and a uniformly random plaintext, respectively. The post-challenge algorithm is also identical for all three adversaries; call it $\mathcal{C}$. The advantage of $\mathcal{B}$ over random guessing is then bounded by

$$\begin{aligned}
&\|\mathcal{C}(\mathsf{Enc}_k(M_0)) - \mathcal{C}(\mathsf{Enc}_k(M_1))\|_1 \\
&\quad = \|\mathcal{C}(\mathsf{Enc}_k(M_0)) - \mathcal{C}(\mathsf{Enc}_k(M_1)) - \mathcal{C}(\mathsf{Enc}_k(R)) + \mathcal{C}(\mathsf{Enc}_k(R))\|_1 \\
&\quad \leq \|\mathcal{C}(\mathsf{Enc}_k(M_0)) - \mathcal{C}(\mathsf{Enc}_k(R))\|_1 + \|\mathcal{C}(\mathsf{Enc}_k(M_1)) - \mathcal{C}(\mathsf{Enc}_k(R))\|_1 \\
&\quad \leq \mathrm{negl}(n)\,,
\end{aligned}$$

where the last inequality follows from our initial assumption, applied to both $\mathcal{B}_0$ and $\mathcal{B}_1$. It follows that Π is IND-QCCA1. ◀

C Proof of Theorem 16

In this appendix, we prove Theorem 16, restated below for convenience.

▶ **Theorem 16.** *Let U_{LRF} be the quantum oracle for the linear rounding function $\mathsf{LRF}_{\boldsymbol{k},a,b}$ with modulus $q \geq 2$, block size $b \in \{1, \ldots, q-1\}$, and an unknown $a \in \{0, \ldots, q-1\}$, and unknown key $\boldsymbol{k} \in \mathbb{Z}_q^n$ such that $\boldsymbol{k}$ has at least one entry that is a unit modulo q. Let $c = \lceil q/b \rceil$ and $d = cb - q$. By making one query to the oracle U_{LRF}, Algorithm 1 recovers the key $\boldsymbol{k}$ with probability at least $4/\pi^2 - O(d/q)$.*

Proof. For an integer m, let $\omega_m = e^{2\pi i/m}$. Several times in this proof, we will make use of the identity $\sum_{z=0}^{\ell-1} \omega_m^{rz} = \omega_m^{r(\ell-1)/2}\left(\frac{\sin(\ell r\pi/m)}{\sin(r\pi/m)}\right)$.

Let $c = \lceil q/b \rceil$. Throughout this proof, let $\mathsf{LRF}(\boldsymbol{x}) = \mathsf{LRF}_{\boldsymbol{k},a,b}(\boldsymbol{x})$. By querying with $\frac{1}{\sqrt{c}}\sum_{z=0}^{c-1}\omega_c^z|z\rangle$ in the second register, we are using the standard phase kickback technique, which puts the output of the oracle directly into the phase:

$$\begin{aligned}
|\boldsymbol{x}\rangle\frac{1}{\sqrt{c}}\sum_{z=0}^{c-1}\omega_c^z|z\rangle \quad &\overset{U_{\mathsf{LRF}}}{\longmapsto} \quad |\boldsymbol{x}\rangle\frac{1}{\sqrt{c}}\sum_{z=0}^{c-1}\omega_c^z|z + \mathsf{LRF}(\boldsymbol{x}) \pmod c\rangle \\
&= \quad |\boldsymbol{x}\rangle\frac{1}{\sqrt{c}}\sum_{z=0}^{c-1}\omega_c^{z-\mathsf{LRF}(\boldsymbol{x})}|z\rangle = \omega_c^{-\mathsf{LRF}(\boldsymbol{x})}|\boldsymbol{x}\rangle\frac{1}{\sqrt{c}}\sum_{z=0}^{c-1}\omega_c^z|z\rangle.
\end{aligned}$$

Thus, after querying the uniform superposition over the cipherspace with $\frac{1}{\sqrt{c}}\sum_{z=0}^{c-1}\omega_c^z|z\rangle$ in the second register, we arrive at the state

$$\frac{1}{\sqrt{q^n}}\sum_{\boldsymbol{x}\in\mathbb{Z}_q^n}\omega_c^{-\mathsf{LRF}(\boldsymbol{x})}|\boldsymbol{x}\rangle\frac{1}{\sqrt{c}}\sum_{z=0}^{c-1}\omega_c^z|z\rangle.$$

Note that $\omega_c = \omega_q^{q/c}$. If we discard the last register and apply $\mathsf{QFT}_{\mathbb{Z}_q}^{\otimes n}$, we get

$$|\psi\rangle = \frac{1}{q^n} \sum_{\boldsymbol{y}\in\mathbb{Z}_q^n} \sum_{\boldsymbol{x}\in\mathbb{Z}_q^n} \omega_q^{-(q/c)\mathsf{LRF}(\boldsymbol{x})+\langle \boldsymbol{x},\boldsymbol{y}\rangle} |\boldsymbol{y}\rangle.$$

We then perform a complete measurement in the computational basis. The probability of obtaining the key $\boldsymbol{k}$ is given by

$$|\langle \boldsymbol{k}|\psi\rangle|^2 = \left| \frac{1}{q^n} \sum_{\boldsymbol{x}\in\mathbb{Z}_q^n} \omega_q^{-\frac{q}{c}\mathsf{LRF}(\boldsymbol{x})+\langle \boldsymbol{x},\boldsymbol{k}\rangle} \right|^2 = \left| \frac{1}{q^n} \sum_{v=0}^{c-1} \omega_q^{-\frac{q}{c}v} \sum_{\boldsymbol{x}\in\mathbb{Z}_q^n:\mathsf{LRF}(\boldsymbol{x})=v} \omega_q^{\langle \boldsymbol{x},\boldsymbol{k}\rangle} \right|^2. \tag{2}$$

We are assuming that $\boldsymbol{k}$ has at least one entry that is a unit modulo q. For simplicity, suppose that entry is k_n. Let $\boldsymbol{k}_{1:n-1}$ denote the first $n-1$ entries of $\boldsymbol{k}$. Then, for any $v \in \{0,\ldots,c-2\}$:

$$\begin{aligned} \sum_{\boldsymbol{x}\in\mathbb{Z}_q^n:\mathsf{LRF}(\boldsymbol{x})=v} \omega_q^{\langle \boldsymbol{x},\boldsymbol{k}\rangle} &= \sum_{\boldsymbol{x}\in\mathbb{Z}_q^n:\langle \boldsymbol{x},\boldsymbol{k}\rangle\in I_v(a,b)} \omega_q^{\langle \boldsymbol{x},\boldsymbol{k}\rangle} \\ &= \sum_{\boldsymbol{y}\in\mathbb{Z}_q^{n-1}} \omega_q^{\langle \boldsymbol{y},\boldsymbol{k}_{1:n-1}\rangle} \sum_{\substack{x_n\in\mathbb{Z}_q:\\ x_nk_n\in I_v(a-\langle \boldsymbol{y},\boldsymbol{k}_{1:n-1}\rangle,b)}} \omega_q^{x_nk_n}. \end{aligned} \tag{3}$$

(Recall the definition of $I_v(a,b)$ from Definition 15). Since k_n is a unit, for each $z \in I_v(a-\langle \boldsymbol{y},\boldsymbol{k}_{1:n-1}\rangle)$, there is a unique $x_n \in \mathbb{Z}_q$ such that $x_nk_n = z$. Thus, for a fixed $\boldsymbol{y}\in\mathbb{Z}_q^{n-1}$, letting $a' = a - \langle \boldsymbol{y},\boldsymbol{k}_{1:n-1}\rangle$, we have:

$$\sum_{x_n\in\mathbb{Z}_q:x_nk_n\in I_v(a',b)} \omega_q^{x_nk_n} = \sum_{z=a'+vb}^{a'+(v+1)b-1} \omega_q^z = \omega_q^{a'+vb} \sum_{z=0}^{b-1} \omega_q^z,$$

which we can plug into (3) to get:

$$\sum_{\substack{\boldsymbol{x}\in\mathbb{Z}_q^n:\\ \mathsf{LRF}(\boldsymbol{x})=v}} \omega_q^{\langle \boldsymbol{x},\boldsymbol{k}\rangle} = \sum_{\boldsymbol{y}\in\mathbb{Z}_q^{n-1}} \omega_q^{\langle \boldsymbol{y},\boldsymbol{k}_{1:n-1}\rangle} \omega_q^{a-\langle \boldsymbol{y},\boldsymbol{k}_{1:n-1}\rangle+vb} \sum_{z=0}^{b-1} \omega_q^z = q^{n-1}\omega_q^{a+vb} \sum_{z=0}^{b-1} \omega_q^z. \tag{4}$$

We can perform a similar analysis for the remaining case when $v = c-1$. Recall that $d = cb - q \geq 0$ so $vb = cb - b = d + q - b = -(b-d) \pmod q$ and we get

$$\sum_{\boldsymbol{x}\in\mathbb{Z}_q^n:\mathsf{LRF}(\boldsymbol{x})=c-1} \omega_q^{\langle \boldsymbol{x},\boldsymbol{k}\rangle} = q^{n-1}\omega_q^{a-(b-d)} \sum_{z=0}^{b-d-1} \omega_q^z. \tag{5}$$

This is slightly different from the $v < c-1$ case, shown in (4), but very similar. If we substitute $v = c-1$ in (4) and compare it to (5), we get

$$\begin{aligned} & \left| q^{n-1}\omega_q^{a-(b-d)} \sum_{z=0}^{b-d-1} \omega_q^z - q^{n-1}\omega_q^{a-(b-d)} \sum_{z=0}^{b-1} \omega_q^z \right| \\ = \; & q^{n-1} \left| \sum_{z=b-d}^{b-1} \omega_q^z \right| = q^{n-1} \left| \sum_{z=0}^{d-1} \omega_q^z \right| = q^{n-1} \left| \frac{\sin(\pi d/q)}{\sin(\pi/q)} \right| \\ \leq \; & q^{n-1} \frac{\pi d/q}{2/q} = q^{n-1}\frac{\pi}{2}d. \end{aligned} \tag{6}$$

Above, we have used the facts $\sin x \leq x$, and $|\sin x| \geq 2x/\pi$ when $|x| \leq \pi/2$. Now, plugging (4) into (2) for all the $v < c-1$ terms, and using (6) and the triangle inequality for the $v = c-1$ term, we get:

$$\begin{aligned} |\langle \boldsymbol{k}|\psi\rangle| &\geq \left| \frac{1}{q^n} \sum_{v=0}^{c-1} \omega_q^{-qv/c} \cdot q^{n-1} \omega_q^{a+vb} \sum_{z=0}^{b-1} \omega_q^z \right| - \left| \frac{1}{q^n} \omega_q^{-q(c-1)/c} \cdot q^{n-1} \frac{\pi}{2} d \right| \\ &= \frac{1}{q} \left| \sum_{v=0}^{c-1} \omega_q^{v(b-q/c)} \frac{\sin(b\pi/q)}{\sin(\pi/q)} \right| - \frac{\pi}{2} \frac{d}{q} \\ &= \frac{1}{q} \frac{\sin(b\pi/q)}{\sin(\pi/q)} \left| \sum_{v=0}^{c-1} \omega_q^{v(b-q/c)} \right| - \frac{\pi}{2} \frac{d}{q}. \end{aligned} \tag{7}$$

Since $b - q/c = d/c$, we can bound the sum as follows:

$$\begin{aligned} \left| \sum_{v=0}^{c-1} \omega_q^{v(b-q/c)} \right| = \left| \sum_{v=0}^{c-1} \omega_q^{vd/c} \right| &\geq \left| \sum_{v=0}^{c-1} \cos\left(\frac{2\pi}{q} \frac{vd}{c} \right) \right| \\ &\geq \left| \sum_{v=0}^{c-1} \cos\left(\frac{2\pi}{q} d \right) \right| = \left| c \cos\left(\frac{2\pi d}{q} \right) \right| \qquad (8) \\ &\geq c\sqrt{1 - (2\pi d/q)^2}. \qquad (9) \end{aligned}$$

To get the inequality (8), we used $0 \leq v \leq c$ and the assumption that $d/q \leq 1/4$ (if $d/q > 1/4$, the claim of the theorem is trivial), which implies that $\frac{2\pi v}{c} \frac{d}{q} \leq \frac{\pi}{2}$. The last inequality follows from $|\cos x| \geq \sqrt{1-x^2}$.

Next, we bound $\frac{\sin(b\pi/q)}{\sin(\pi/q)}$. When $b/q \leq 1/2$, $b\pi/q \leq \pi/2$, so we have $\sin(b\pi/q) \geq 2b/q$. We also have $\sin(\pi/q) \leq \pi/q$. Thus,

$$\frac{\sin(b\pi/q)}{\sin(\pi/q)} \geq \frac{2b}{\pi}.$$

On the other hand, when $b/q > 1/2$, we must have $c = 2$ and $b = \frac{q+d}{2}$. In that case

$$\sin(b\pi/q) = \sin \frac{\pi(q+d)}{2q} = \sin\left(\frac{\pi}{2} + \frac{\pi}{2} \frac{d}{q} \right) = \cos \frac{\pi d}{2q} \geq \sqrt{1 - \left(\frac{\pi d}{2q} \right)^2}.$$

Since $\sin(\pi/q) \leq \pi/q$ and $q \geq 2b$,

$$\frac{\sin(b\pi/q)}{\sin(\pi/q)} \geq \frac{\sqrt{1 - \left(\frac{\pi d}{2q} \right)^2}}{\pi/q} \geq \frac{2b}{\pi} \sqrt{1 - O(d/q)}.$$

Thus, in both cases, $\frac{\sin(b\pi/q)}{\sin(\pi/q)} \geq \frac{2b}{\pi}\sqrt{1-O(d/q)}$. Plugging this and (9) into (7), we get:

$$\begin{aligned} |\langle \boldsymbol{k}, \psi\rangle| &\geq \frac{1}{q} \cdot \frac{2b}{\pi} \sqrt{1 - O(d/q)} \cdot c\sqrt{1 - O(d/q)} - O(d/q) \\ &= \frac{2}{\pi} \frac{bc}{q} - O(d/q) = \frac{2}{\pi} \frac{q+d}{q} - O(d/q) = \frac{2}{\pi} - O(d/q), \end{aligned}$$

completing the proof. ◀

D Proof of Theorem 22

In this appendix, we prove Theorem 22, restated below for convenience.

▶ **Theorem 22.** *There exists a quantum algorithm that makes one quantum query to* $\mathsf{FrodoPKE.Dec}_{\boldsymbol{S}}$ *and recovers any choice of* $\bar{m}$ *of the* $\bar{n}$ *columns of* $\boldsymbol{S}$*. For each of the chosen columns, if that column has at least one odd entry, then the algorithm succeeds in recovering the column with probability at least* $4/\pi^2$*.*

Proof. Let $\boldsymbol{s}^1, \ldots, \boldsymbol{s}^{\bar{n}}$ be the columns of $\boldsymbol{S}$. Let U denote the map:

$$U : |\boldsymbol{c}\rangle|z_1\rangle \ldots |z_{\bar{n}}\rangle \mapsto |\boldsymbol{c}\rangle|z_1 + \mathsf{LRF}_{\boldsymbol{s}^1,0,q/2^B}(\boldsymbol{c})\rangle \ldots |z_{\bar{n}} + \mathsf{LRF}_{\boldsymbol{s}^{\bar{n}},0,q/2^B}(\boldsymbol{c})\rangle,$$

for any $\boldsymbol{c} \in \mathbb{Z}_q^n$ and $z_1, \ldots, z_{\bar{n}} \in \mathbb{Z}_{2^B}$. We first argue that one call to $\mathsf{FrodoKEM.Dec}_{\boldsymbol{S}}$ can be used to implement $U^{\otimes \bar{m}}$. Then we show that one call to U can be used to recover any choice of the columns of $\boldsymbol{S}$ with probability $4/\pi^2$, as long as it has at least one entry that is odd.

Let $\mathsf{Trunc} : \mathbb{Z}_q \mapsto \mathbb{Z}_{2^B}$ denote the map that takes $x \in \mathbb{Z}_q$ to the integer represented by the B most significant bits of the binary representation of x. We have, for any $\boldsymbol{C}_1 \in \mathbb{Z}_q^{\bar{m}\times n}$, $\boldsymbol{C}_2 = 0^{\bar{m}\times\bar{n}}$, and any $\{z_{i,j}\}_{i\in[\bar{m}],j\in[\bar{n}]} \subseteq \mathbb{Z}_{2^B}$:

$$U_{\mathsf{FrodoKEM.Dec}} : |\boldsymbol{C}_1\rangle|0^{\bar{m}\cdot\bar{n}}\rangle \bigotimes_{i\in[\bar{m}],j\in[\bar{n}]} |z_{i,j}\rangle \mapsto |\boldsymbol{C}_1\rangle|0^{\bar{m}\cdot\bar{n}}\rangle \bigotimes_{i\in[\bar{m}],j\in[\bar{n}]} |z_{i,j} + \mathsf{Trunc}([\boldsymbol{C}_1\boldsymbol{S}]_{i,j})\rangle. \tag{10}$$

Above, $[\boldsymbol{C}_1\boldsymbol{S}]_{i,j}$ represents the ij-th entry of $\boldsymbol{C}_1\boldsymbol{S}$. If $\boldsymbol{c}^1, \ldots, \boldsymbol{c}^{\bar{m}}$ denote the rows of $\boldsymbol{C}_1$, then $[\boldsymbol{C}_1\boldsymbol{S}]_{i,j} = \langle \boldsymbol{c}^i, \boldsymbol{s}^j \rangle$. Thus, $\mathsf{Trunc}([\boldsymbol{C}_1\boldsymbol{S}]_{i,j}) = \mathsf{LRF}_{\boldsymbol{s}^j,0,q/2^B}(\boldsymbol{c}^i)$, the linear rounding function with block size $b = q/2^B$, which is an integer since q is a power of 2, and $a = 0$. Note that we have also assumed that the plaintext is *subtracted* rather than added to the last register; this is purely for convenience of analysis, and can easily be accounted for by adjusting Algorithm 1 (e.g., by using inverse-QFT instead of QFT.)

Discarding the second register (containing $\boldsymbol{C}_2 = 0$), the right-hand side of (10) becomes

$$|\boldsymbol{c}^1\rangle \ldots |\boldsymbol{c}^{\bar{m}}\rangle \bigotimes_{i\in[\bar{m}],j\in[\bar{n}]} |z_{i,j} + \mathsf{LRF}_{\boldsymbol{s}^j,0,q/2^B}(\boldsymbol{c}^i)\rangle. \tag{11}$$

Reordering the registers of (11), we get:

$$\bigotimes_{i\in[\bar{m}]} \left(|\boldsymbol{c}^i\rangle \bigotimes_{j\in[\bar{n}]} |z_{i,j} + \mathsf{LRF}_{\boldsymbol{s}^j,0,q/2^B}(\boldsymbol{c}^i)\rangle \right) = U^{\otimes\bar{m}} \left(\bigotimes_{i\in[\bar{m}]} |\boldsymbol{c}^i\rangle \bigotimes_{j\in[\bar{n}]} |z_{i,j}\rangle \right).$$

Thus, we can implement $U^{\otimes\bar{m}}$ using a single call to $\mathsf{FrodoKEM.Dec}_{\boldsymbol{S}}$.

Next we show that for any particular $j \in [\bar{n}]$, a single call to U can be used to recover $\boldsymbol{s}^j$, the j-th column of $\boldsymbol{S}$, with probability at least $4/\pi^2$, as long as at least one entry of $\boldsymbol{s}^j$ is odd. To do this, we show how one use of U can be used to implement one phase query to $\mathsf{LRF}_{\boldsymbol{s}^j,0,q/2^B}$. Then the result follows from the proof of Theorem 16.

Let $|\varphi\rangle = 2^{-B/2} \sum_{z=0}^{2^B-1} |z\rangle$, and define

$$|\phi_j\rangle = |\varphi\rangle^{\otimes(j-1)} \otimes \frac{1}{\sqrt{2^B}} \sum_{z=0}^{2^B-1} \omega_{2^B}^z |z\rangle \otimes |\varphi\rangle^{\otimes(\bar{n}-j)}.$$

Then for any $\boldsymbol{c} \in \mathbb{Z}_q^n$, we have:

$$\frac{1}{\sqrt{2^B}} \sum_{z=0}^{2^B-1} |z + \mathsf{LRF}_{\boldsymbol{s}^i,0,q/2^B}(\boldsymbol{c})\rangle = \frac{1}{\sqrt{2^B}} \sum_{z=0}^{2^B-1} |z\rangle = |\varphi\rangle,$$

since addition here is modulo 2^B, and

$$\frac{1}{\sqrt{2^B}}\sum_{z=0}^{2^B-1}\omega_{2^B}^{z}|z+\mathsf{LRF}_{\boldsymbol{s}^j,0,q/2^B}(\boldsymbol{c})\rangle=\frac{1}{\sqrt{2^B}}\sum_{z=0}^{2^B-1}\omega_{2^B}^{z-\mathsf{LRF}_{\boldsymbol{s}^j,0,q/2^B}(\boldsymbol{c})}|z\rangle.$$

Thus:

$$\begin{aligned}U(|\boldsymbol{c}\rangle|\phi_j\rangle) &= |\boldsymbol{c}\rangle|\varphi\rangle^{\otimes(j-1)}\otimes\frac{1}{\sqrt{2^B}}\sum_{z=0}^{2^B-1}\omega_{2^B}^{z-\mathsf{LRF}_{\boldsymbol{s}^j,0,q/2^B}(\boldsymbol{c})}|z\rangle\otimes|\varphi\rangle^{\otimes(\bar{n}-j)}\\ &= \omega_{2^B}^{-\mathsf{LRF}_{\boldsymbol{s}^j,0,q/2^B}(\boldsymbol{c})}|\boldsymbol{c}\rangle|\phi_j\rangle.\end{aligned}$$

Thus, by the proof of Theorem 16, if we apply U to $q^{-n/2}\sum_{\boldsymbol{c}\in\mathbb{Z}_q^n}|\boldsymbol{c}\rangle|\phi_j\rangle$, Fourier transform the first register, and then measure, assuming $\boldsymbol{s}^j$ has at least one entry that is a unit[3] we will measure $\boldsymbol{s}^j$ with probability at least $\pi^2/4-O(d/q)$, where $d=q/2^B\lceil q/(q/2^B)\rceil-q=0$.

Thus, if we want to recover columns $j_1,\ldots j_{\bar{m}}$ of $\boldsymbol{S}$, we apply our procedure for $U^{\otimes\bar{m}}$, which costs one query to $\mathsf{FrodoKEM.Dec}_{\boldsymbol{S}}$, to the state

$$\sum_{\boldsymbol{c}\in\mathbb{Z}_q^n}\frac{1}{\sqrt{q^n}}|\boldsymbol{c}\rangle|\phi_{j_1}\rangle\otimes\cdots\otimes\sum_{\boldsymbol{c}\in\mathbb{Z}_q^n}\frac{1}{\sqrt{q^n}}|\boldsymbol{c}\rangle|\phi_{j_{\bar{m}}}\rangle,$$

Fourier transform each of the $\boldsymbol{c}$ registers, and then measure. ◀

[3] since q is a power of 2, this is just an odd number

Quantum Distinguishing Complexity, Zero-Error Algorithms, and Statistical Zero Knowledge

Shalev Ben-David
University of Waterloo, Waterloo, ON, Canada
shalev.b@uwaterloo.ca

Robin Kothari
Quantum Architectures and Computation (QuArC) group, Microsoft Research, Redmond, WA, USA
robin.kothari@microsoft.com

Abstract

We define a new query measure we call quantum distinguishing complexity, denoted $\mathrm{QD}(f)$ for a Boolean function f. Unlike a quantum query algorithm, which must output a state close to $|0\rangle$ on a 0-input and a state close to $|1\rangle$ on a 1-input, a "quantum distinguishing algorithm" can output any state, as long as the output states for any 0-input and 1-input are distinguishable.

Using this measure, we establish a new relationship in query complexity: For all total functions f, $\mathrm{Q}_0(f) = \widetilde{O}(\mathrm{Q}(f)^5)$, where $\mathrm{Q}_0(f)$ and $\mathrm{Q}(f)$ denote the zero-error and bounded-error quantum query complexity of f respectively, improving on the previously known sixth power relationship.

We also define a query measure based on quantum statistical zero-knowledge proofs, $\mathrm{QSZK}(f)$, which is at most $\mathrm{Q}(f)$. We show that $\mathrm{QD}(f)$ in fact lower bounds $\mathrm{QSZK}(f)$ and not just $\mathrm{Q}(f)$. $\mathrm{QD}(f)$ also upper bounds the (positive-weights) adversary bound, which yields the following relationships for all f: $\mathrm{Q}(f) \geq \mathrm{QSZK}(f) \geq \mathrm{QD}(f) = \Omega(\mathrm{Adv}(f))$. This sheds some light on why the adversary bound proves suboptimal bounds for problems like Collision and Set Equality, which have low QSZK complexity.

Lastly, we show implications for lifting theorems in communication complexity. We show that a general lifting theorem for either zero-error quantum query complexity or for QSZK would imply a general lifting theorem for bounded-error quantum query complexity.

2012 ACM Subject Classification Theory of computation → Quantum complexity theory

Keywords and phrases Quantum query complexity, quantum algorithms

Digital Object Identifier 10.4230/LIPIcs.TQC.2019.2

Related Version A full version of the paper is available at `https://arxiv.org/abs/1902.03660`.

Acknowledgements We thank Scott Aaronson, Mika Göös, John Watrous, and Ronald de Wolf for helpful conversations about this work. Most of this work was performed while the first author was at the Massachusetts Institute of Technology and the University of Maryland and the second author was at the Massachusetts Institute of Technology. This work was partially supported by NSF grant CCF-1629809.

1 Introduction

In the model of query complexity, we wish to compute some known Boolean function $f : \{0,1\}^n \to \{0,1\}$ on an unknown input $x \in \{0,1\}^n$ that we can access through an oracle that knows x. In the classical setting, the oracle responds with x_i when queried with an index $i \in [n]$. For quantum models, we use essentially the same oracle, but slightly modified to make it unitary. The bounded-error quantum query complexity of a function f, denoted $\mathrm{Q}(f)$, is the minimum number of queries to the oracle needed to compute the function f with probability greater than $2/3$ on any input x. In other words, the quantum query algorithm outputs a quantum state that is close to $|f(x)\rangle$.

In this paper we study "quantum distinguishing complexity," a query measure obtained by relaxing the output requirement of quantum query algorithms. Essentially, a quantum distinguishing algorithm for f doesn't need to compute $f(x)$, but merely needs to *behave differently on input x and input y if $f(x) \neq f(y)$*. We claim that this weaker notion of computation helps shed light on quantum query complexity and various lower bound techniques for it. We use quantum distinguishing complexity to prove a new query complexity relationship for total functions: $\mathrm{Q}_0(f) = O(\mathrm{Q}(f)^5 \log \mathrm{Q}(f))$. We also use it to explain why the non-negative adversary bound fails for some problems, to provide lower bound techniques for the query version of the complexity class QSZK, and to prove some reductions between lifting theorems in communication complexity.

1.1 Quantum distinguishing complexity

The *quantum distinguishing complexity* of a function $f : D \to \{0,1\}$ (where $D \subseteq \{0,1\}^n$), denoted $\mathrm{QD}(f)$, is the minimum number of queries needed to the input $x \in D$ to produce an output state $|\psi_x\rangle$, such that the output states corresponding to 0-inputs and 1-inputs are nearly orthogonal (or far apart in trace distance). Note that the usual bounded-error quantum query complexity of a function f, denoted $\mathrm{Q}(f)$, is defined similarly with the additional requirement that there should exist a 2-outcome measurement that (with high probability) accepts states corresponding to 1-inputs and rejects states corresponding to 0-inputs. Since measurements can only distinguish nearly orthogonal states, every quantum algorithm for computing f satisfies the definition of quantum distinguishing complexity. Hence for all functions f, we have $\mathrm{QD}(f) \leq \mathrm{Q}(f)$. We formally define quantum distinguishing complexity and establish some basic properties in Section 3.

This is a natural relaxation of bounded-error quantum query complexity and has been mentioned in passing in several prior works. Indeed, Barnum, Saks, and Szegedy call this measure $\mathrm{DQA}(f)$ in an early technical report [7, Remark 1]. This measure often comes up in discussions about the (positive-weights) adversary bound,[1] a lower bound for quantum query complexity introduced by Ambainis [4]. The (positive-weights) adversary bound, which we denote by $\mathrm{Adv}(f)$, has several variants [4, 5, 8, 23, 37], which are all essentially the same [32]. It was noted in several works [8, 21] that the proof that the adversary bound lower bounds quantum query complexity only uses the fact that the outputs corresponding to 0-inputs and 1-inputs are nearly orthogonal, and hence for all functions $\mathrm{QD}(f) = \Omega(\mathrm{Adv}(f))$. However, it is not the case that $\mathrm{QD}(f) = \Theta(\mathrm{Adv}(f))$ for all f, and we exhibit functions separating these measures.

Lastly, we show in Section 3 that this measure is the quantum analogue of a lower bound method for randomized query complexity called randomized sabotage complexity [11]. Hence this measure could also be called "quantum sabotage complexity."

1.2 Fifth power query relation

Our first result establishes a new relation between query measures for total functions. A total function is a function of the form $f : \{0,1\}^n \to \{0,1\}$, as opposed to a partial function, which is a function of the form $f : D \to \{0,1\}$, where $D \subseteq \{0,1\}^n$. We show a new upper bound on the zero-error quantum query complexity of f, denoted $\mathrm{Q}_0(f)$, in terms of its quantum

[1] The positive-weights adversary bound should not be confused with the stronger negative-weights adversary bound (also known as the general adversary bound), which essentially equals quantum query complexity [21, 24].

distinguishing complexity, and hence its quantum query complexity. The zero-error quantum query complexity of f is the minimum number of queries needed by a quantum algorithm that either outputs the correct answer $f(x)$ on input x, or outputs ? indicating that it does not know, but does this with probability at most $1/2$ on any input x. In Section 4 we prove the following.

▶ **Theorem 1.** *For all total functions $f : \{0,1\}^n \to \{0,1\}$, we have*

$$\mathrm{Q}_0(f) = O(\mathrm{QD}(f)^5 \log \mathrm{QD}(f)) = O(\mathrm{Q}(f)^5 \log \mathrm{Q}(f)). \tag{1}$$

Additionally, the algorithm also outputs a certificate for $f(x)$ when it outputs $f(x)$.

This is an improvement over the previous best relationship between zero-error and bounded-error quantum query complexity, $\mathrm{Q}_0(f) = O(\mathrm{Q}(f)^6)$ [9], which follows from $\mathrm{D}(f) = O(\mathrm{Q}(f)^6)$, where $\mathrm{D}(f)$ is deterministic query complexity. In fact, our result is the first upper bound on zero-error quantum query complexity that does not follow from an upper bound on zero-error randomized query complexity. Our proof borrows ideas from the classical result $\mathrm{R}_0(f) = O(\mathrm{R}(f)^2 \log \mathrm{R}(f))$ [27, 22], which is essentially optimal due to a nearly matching separation by Ambainis et al. [6].

1.3 Quantum statistical zero knowledge

Next we show that, surprisingly, quantum distinguishing complexity lower bounds a more powerful model of computation than quantum query complexity: the query complexity of computing a function using a quantum statistical zero-knowledge (QSZK) proof system. A QSZK proof system is an interactive protocol between a quantum verifier and a computationally unbounded, but untrusted prover in which the verifier learns the value of $f(x)$ but learns essentially no more. QSZK can also be characterized in terms of its complete problem Quantum State Distinguishability [34, 35].

In Section 5, we discuss the history of quantum statistical zero-knowledge proofs and define an associated query measure $\mathrm{QSZK}(f)$ based on the complete problem Quantum State Distinguishability. We establish some basic properties of our definition, such as $\mathrm{QSZK}(f) \leq \mathrm{Q}(f)$, which corresponds to the complexity class containment $\mathsf{BQP} \subseteq \mathsf{QSZK}$. We then show that quantum distinguishing complexity lower bounds QSZK complexity.

▶ **Theorem 2.** *For all (partial) Boolean functions f, $\mathrm{QD}(f) \leq \mathrm{QSZK}(f)$.*

As a corollary of Theorem 2 and $\mathrm{QD}(f) = \Omega(\mathrm{Adv}(f))$, we have for all (partial) functions f,

$$\mathrm{Q}(f) \geq \mathrm{QSZK}(f) \geq \mathrm{QD}(f) = \Omega(\mathrm{Adv}(f)). \tag{2}$$

This sheds some light on why the adversary bound sometimes proves poor lower bounds: it lower bounds a more powerful model of computation! For example, it is well known that the adversary bound cannot prove a super-constant lower bound for the collision problem [3]. It is also easy to see that the collision problem has a constant-query QSZK (and even classical SZK) protocol.

On the bright side, this gives us a new way to prove lower bounds on QSZK query complexity and prove oracle separations against the complexity class QSZK. For example, since we know the OR function on n bits has $\mathrm{Adv}(\mathrm{OR}) = \Omega(\sqrt{n})$, this yields an oracle A such that $\mathsf{NP}^A \not\subseteq \mathsf{QSZK}^A$, since the OR function has small certificates. A similar strategy was used recently by Menda and Watrous to show oracle separations against QSZK [26].

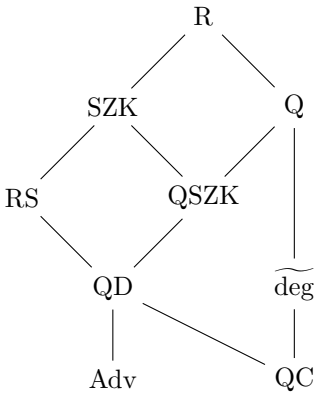


Figure 1 Relationships between measures. An upward line indicates that a measure is asymptotically upper bounded by the other measure. E.g., for all (partial) functions f, $\mathrm{Q}(f) = O(\mathrm{R}(f))$.

1.4 Comparison with other lower bounds

We compare quantum distinguishing complexity to the two main lower bound techniques for quantum query complexity: the (positive-weights) adversary bound and the polynomial method. Recall that the negative-weights adversary or general adversary completely characterizes quantum query complexity, so we do not compare quantum distinguishing complexity with it.

As noted earlier, the adversary bound is weaker than quantum distinguishing complexity since for all (partial) functions f, $\mathrm{QD}(f) = \Omega(\mathrm{Adv}(f))$. This implies that $\mathrm{QD}(f)$ coincides with $\mathrm{Q}(f)$ for most functions studied in the literature, since most quantum lower bounds are proved using the adversary method. Moreover, not only is quantum distinguishing complexity always larger than the adversary bound, it can be exponentially larger for partial functions and quadratically larger for total functions as we show in Theorem 3.

Another popular lower bound technique is the polynomial method [9], which uses the fact that the approximate degree of a function lower bounds $\mathrm{Q}(f)$. The approximate degree of a Boolean function f, denoted $\widetilde{\deg}(f)$, is the minimum degree of a real polynomial $p(x)$ over the input variables such that for all inputs x we have $|f(x) - p(x)| \leq 1/3$.

We do not know an exponential separation between quantum distinguishing complexity and approximate degree (for a partial function), since it is not even known if quantum query complexity can be exponentially larger than approximate degree for a partial function. We do, however, show in Theorem 3 that quantum distinguishing complexity can be polynomially larger than approximate degree for total functions.

▶ **Theorem 3.** *There exist total functions f and g with* $\mathrm{QD}(f) = \widetilde{\Omega}(\mathrm{Adv}(f)^2)$ and $\mathrm{QD}(g) \geq \widetilde{\deg}(g)^{4-o(1)}$.
There also exists an n-bit partial function h with $\mathrm{QD}(h) = \widetilde{\Omega}(n^{1/3})$ and $\mathrm{Adv}(h) = O(\log n)$.

This theorem is proved in Section 6. Figure 1 shows the known relationships between all the measures discussed in this paper. The measures RS and QC are introduced later, and refer to randomized sabotage complexity and quantum certificate complexity, respectively.

1.5 Lifting theorems

Most measures in query complexity have an analogous measure in communication complexity, which we denote with the superscript cc, such as $\mathrm{Q}^{\mathrm{cc}}(F)$ and $\mathrm{QSZK}^{\mathrm{cc}}(F)$. A lifting theorem is a result that transfers a lower bound on a query function f to a lower bound in communication

complexity for a lifted version of the function f, obtained by composing the function f with a hard communication problem G. For example, a lifting theorem is known for deterministic protocols, which means there exists a communication problem G such that for all functions f, $\mathrm{D}^{\mathrm{cc}}(f \circ G) = \Omega(\mathrm{D}(f))$ [29, 19].

Lifting theorems have been shown for some measures, such as nondeterministic query complexity [18] and (zero-error or bounded-error) randomized query complexity [20], and remain open for measures like zero-error and bounded-error quantum query complexity. Our next result, proved in Section 7, shows that if we could prove a lifting theorem for zero-error quantum query complexity or for QSZK query complexity, then we would get a lifting theorem for bounded-error quantum query complexity.

▶ **Theorem 4** (informal). *If a general lifting theorem holds using some gadget G for either zero-error quantum query complexity, i.e., $\mathrm{Q}_0^{\mathrm{cc}}(f \circ G) = \widetilde{\Omega}(\mathrm{Q}_0(f))$, or for quantum statistical zero-knowledge protocols, i.e., $\mathrm{QSZK}^{\mathrm{cc}}(f \circ G) = \widetilde{\Omega}(\mathrm{QSZK}(f))$, then we obtain a general lifting theorem for bounded-error quantum query complexity (up to logarithmic factors) with the same gadget G.*

In fact, the same conclusion follows from a weaker assumption. We can assume that the lifting theorem proves a lower bound on bounded-error quantum communication complexity assuming a lower bound on quantum distinguishing complexity. In other words, we can assume a lifting theorem of the form $\mathrm{Q}^{\mathrm{cc}}(f \circ G) = \widetilde{\Omega}(\mathrm{QD}(f))$, which is weaker than a QSZK lifting theorem since it assumes a stronger lower bound and proves a weaker one.

2 Preliminaries

We assume the reader is generally familiar with quantum computation [28] and query complexity (for more details, see [15]). We do not assume the reader is familiar with statistical zero-knowledge protocols.

For any positive integer n, let $[n] = \{1, \ldots, n\}$. We use $f(n) = \widetilde{O}(g(n))$ to mean there exists a constant k such that $f(n) = O(g(n) \log^k g(n))$ and similarly $f(n) = \widetilde{\Omega}(g(n))$ means $f(n) = \Omega(g(n)/\log^k g(n))$ for some constant k.

2.1 Distance measures

For any matrix A, we define the spectral norm of A, denoted $\|A\|$ as the largest singular value of A. The 1-norm of A, denoted $\|A\|_1$, is defined as $\mathrm{Tr}\left(\sqrt{A^\dagger A}\right)$, which is also equal to the sum of the singular values of A.

We define the trace distance between two quantum states ρ and σ as $\|\rho - \sigma\|_{\mathrm{tr}} = \frac{1}{2}\|\rho - \sigma\|_1$. The factor of $1/2$ makes this distance measure lie between 0 and 1 for density matrices. Trace distance is a useful distance measure since it exactly captures distinguishability of states and is non-increasing under quantum operations [28, Th. 9.2]. For pure states $|\psi\rangle$ and $|\phi\rangle$, trace distance is related to their inner product as follows [36, eq. 1.186].

$$\||\psi\rangle\langle\psi| - |\phi\rangle\langle\phi|\|_{\mathrm{tr}} = \sqrt{1 - |\langle\psi|\phi\rangle|^2}. \tag{3}$$

2.2 Quantum query complexity

In query complexity, we wish to compute a Boolean function f on an input x given query access to the bits of x. In this paper, we will mostly deal with functions with Boolean input and output. An n-bit function $f : \{0,1\}^n \to \{0,1\}$ is called a *total function*. An n-bit

function $f : D \to \{0,1\}$, where $D \subseteq \{0,1\}^n$, is called a *partial function* since it is defined on a subset of $\{0,1\}^n$. We will also refer to this subset D as the domain of f, or $\mathrm{Dom}(f)$. The goal in query complexity is to compute $f(x)$ while making the fewest queries to the oracle for the bits of x.

Classical algorithms have access to an oracle that given an index $i \in [n]$ outputs x_i, the i^{th} bit of x. A quantum algorithm is allowed access to a unitary map that implements this oracle, and is usually taken to be the unitary O_x which acts as follows on inputs $i \in [n]$ and $b \in \{0,1\}$: $O_x|i,b\rangle = |i, b \oplus x_i\rangle$. A quantum algorithm that uses the gate O_x in its circuit k times is said to have made k queries to the oracle.

Since we do not count the complexity of any other gates used in the algorithm, we can assume a k-query quantum algorithm always starts with the all-zeros state $|0^m\rangle$ and applies an oracle-independent unitary U_0 followed by the oracle O_x and so on. Thus a k-query quantum algorithm is specified by $k+1$ oracle-independent unitaries $U_0, \ldots, U_k$, which act on m output qubits. The state output by the quantum algorithm is $|\psi_x\rangle = U_k O_x U_{k-1} O_x \cdots O_x U_1 O_x U_0 |0^m\rangle$, where O_x is implicitly $(O_x \otimes \mathbb{1})$ if U_i acts on more qubits than O_x. If the quantum algorithm outputs a mixed state, then we assume it traces out some subset S of the m qubits, and hence outputs $\mathrm{Tr}_S(|\psi_x\rangle\langle\psi_x|)$. If the quantum algorithm outputs a bit, then we assume it measures the first qubit in the standard basis and outputs the result of that measurement.

We can now define the various complexity measures associated with quantum query complexity. We say the *bounded-error quantum query complexity* of computing a Boolean function f, $\mathrm{Q}(f)$, is the minimum k such that there exists a k-query quantum algorithm that on every $x \in \mathrm{Dom}(f)$ outputs $f(x)$ with probability greater than or equal to $2/3$. As usual, the constant $2/3$ is unimportant as long as it is a constant strictly greater than half, due to standard error reduction.

A *zero-error quantum algorithm* (or a Las Vegas quantum algorithm) never outputs an incorrect answer on an input $x \in \mathrm{Dom}(f)$, but is allowed to claim ignorance and answer `?` with probability at most $1/2$. The *zero-error quantum query complexity* of f, $\mathrm{Q}_0(f)$ is the minimum number of queries needed for a zero-error quantum algorithm to compute f. Note that $\mathrm{Q}(f) \leq \mathrm{Q}_0(f)$, since a zero-error algorithm can be turned into a bounded-error algorithm by simply outputting a random bit when the zero-error algorithm outputs `?`.

For zero-error quantum algorithms, there is a subtlety to do with whether or not the algorithm also produces a classical certificate for the input x. A certificate for x is a subset of bits of x, such that the value of $f(x)$ is completely determined by reading these bits alone. A classical zero-error algorithm can always be assumed to output such a certificate without loss of generality. However, this is not known to be true for zero-error quantum algorithms, and zero-error quantum algorithms that also output a certificate when they output a non-`?` answer are called *self-certifying* algorithms [14]. All the zero-error quantum algorithms in this paper are self-certifying, which makes our results stronger since we only prove upper bounds on zero-error quantum algorithms.

3 Quantum distinguishing complexity

3.1 Definition

We now define quantum distinguishing complexity more formally. As explained in the introduction, instead of requiring that the quantum algorithm output the value of the function $f(x)$, as in standard quantum query complexity, we only want the quantum algorithm's outputs to be distinguishable (or nearly orthogonal) for 0-inputs and 1-inputs.

As an example of how these definitions differ, consider the collision problem. In this problem, we are given an input $x \in [n]^n$ and we are promised that if we view x as a function from $[n] \to [n]$, the function is either 1-to-1 or 2-to-1. The goal is to distinguish these two cases under the assumption that the input satisfies this promise. In this problem, since every 0-input and 1-input differ in exactly half the positions $i \in [n]$, our quantum algorithm can simply create the state $|\psi_x\rangle = \frac{1}{\sqrt{n}} \sum_i |i, x_i\rangle$ and the states corresponding to 0-inputs and 1-inputs will have trace distance $\Omega(1)$. Thus this problem has quantum distinguishing complexity $O(1)$, but its quantum query complexity is $\Theta(n^{1/3})$ [3].

▶ **Definition 5** (Quantum Distinguishing complexity). *Let $f : D \to \{0,1\}$, where $D \subseteq \{0,1\}^n$, be an n-bit partial function. $\mathrm{QD}(f)$ is defined as the smallest integer k such that there exists a k-query quantum algorithm that on input $x \in D$ outputs a quantum state ρ_x such that*

$$\forall x, y \in D \text{ with } f(x) \neq f(y), \quad \|\rho_x - \rho_y\|_{\mathrm{tr}} \geq 1/6. \tag{4}$$

Note that the definition is robust to minor changes. First, we allow outputting mixed states, although this does not offer any additional power over only outputting pure states. The reason is that we can always assume that the quantum algorithm is pure until the final step where some subset of qubits is traced out. But if two states are far apart in trace distance after a partial trace, then they were far apart to begin with since trace distance is non-increasing under partial trace.

The constant $1/6$ in Definition 5 is also arbitrary and any constant in $(0, 1)$ would not change the measure by more than a multiplicative constant. This is because we can increase the trace distance between the states by outputting multiple copies of the states. We choose the constant $1/6$ purely for aesthetic reasons: This choice ensures that the result in Theorem 2 has no constant factors.

3.2 Properties

We can now establish some basic properties of quantum distinguishing complexity. First, let us formally show that quantum distinguishing complexity lower bounds quantum query complexity.

▶ **Proposition 6.** *For all (partial) Boolean functions f, $\mathrm{QD}(f) \leq \mathrm{Q}(f)$.*

Proof. Let $\mathrm{Q}(f) = k$ and consider the k-query algorithm that witnesses this fact. Let p_x be the probability that this k-query algorithm, when run on input x, outputs 1 upon measuring the first qubit. Since the algorithm computes f with bounded error, we know that for all 1-inputs x, $p_x \geq 2/3$, and for all 0-inputs y, $p_y \leq 1/3$.

Now consider the single-qubit state ρ_x, which is obtained by taking the final state of this algorithm, tracing out all the qubits except the first one, and then applying a completely dephasing channel to it. This state is $\rho_x = \left(\begin{smallmatrix} 1-p_x & 0 \\ 0 & p_x \end{smallmatrix}\right)$. Thus for all x, y with $f(x) \neq f(x)$, $\|\rho_x - \rho_y\|_{\mathrm{tr}} = |p_x - p_y| \geq 1/3$. ◀

As noted in the introduction, quantum distinguishing complexity is also lower bounded by the adversary bound, i.e., $\mathrm{QD}(f) = \Omega(\mathrm{Adv}(f))$.

We do not prove this since this follows from the arguments that establish that the adversary bound is a lower bound on quantum query complexity [4, 5, 8, 23, 37, 32], since all these proofs only use the fact that the states output on 0-inputs and 1-inputs are nearly orthogonal.

Quantum distinguishing complexity is also superior to quantum certificate complexity $\mathrm{QC}(f)$, as we show in Proposition 8. Quantum certificate complexity is a lower bound on quantum query complexity defined by Aaronson [1]. It was later shown that quantum certificate complexity also lower bounds approximate polynomial degree [22].

Before proving Proposition 8, we first define certificate complexity, randomized certificate complexity, and quantum certificate complexity.

▶ **Definition 7** (Certificate complexity). *For any (partial) function f and input $x \in \mathrm{Dom}(f)$, consider the partial function f^x defined on the domain $\{x\} \cup \{y \in \mathrm{Dom}(f) : f(y) \neq f(x)\}$ that satisfies $f^x(x) = 1$ and $f^x(y) = 0$ for all $y \in \mathrm{Dom}(f)$ with $f(y) \neq f(x)$.*

We define the certificate complexity of f, denoted $\mathrm{C}(f)$, the randomized certificate complexity of f, denoted $\mathrm{RC}(f)$, and the quantum certificate complexity of f, denoted $\mathrm{QC}(f)$, as follows:

$$\mathrm{C}(f) = \max_{x \in \mathrm{Dom}(f)} \mathrm{D}(f^x), \quad \mathrm{RC}(f) = \max_{x \in \mathrm{Dom}(f)} \mathrm{R}(f^x), \quad \text{and} \quad \mathrm{QC}(f) = \max_{x \in \mathrm{Dom}(f)} \mathrm{Q}(f^x). \tag{5}$$

The problem f^x is clearly no harder than computing f itself in any model of computation, and hence these are lower bounds on their respective measures, i.e., $\mathrm{C}(f) \leq \mathrm{D}(f)$, $\mathrm{RC}(f) \leq \mathrm{R}(f)$, and $\mathrm{QC}(f) \leq \mathrm{Q}(f)$. We can now prove that $\mathrm{QD}(f)$ is a better lower bound on $\mathrm{Q}(f)$ than $\mathrm{QC}(f)$.

▶ **Proposition 8.** *For all (partial) Boolean functions f, $\mathrm{QD}(f) = \Omega(\mathrm{QC}(f))$.*

Proof. Let $\mathrm{QD}(f) = k$ and consider the k-query quantum algorithm that witnesses this fact. We can use this algorithm to solve f^x for any $x \in \mathrm{Dom}(f)$. Consider the output of the algorithm on input x before the partial trace operation and call this $|\psi_x\rangle$. The trace distance between $|\psi_x\rangle$ and $|\psi_y\rangle$ for $y \in \mathrm{Dom}(f)$ with $f(y) \neq f(x)$ is at least $1/6$ since trace distance is non-increasing under partial trace [28, Th. 9.2].

Now we construct an algorithm for f^x from this algorithm to show that $\mathrm{Q}(f^x) = O(\mathrm{QD}(f))$. To do so, we run the supposed algorithm and measure whether the output state is $|\psi_x\rangle$ or not and accept only when the measurement accepts. This yields an algorithm that outputs 1 on x with probability 1 and accepts inputs y with $f(x) \neq f(x)$ with some constant probability strictly less than 1. More precisely, the acceptance probability is $|\langle\psi_x|\psi_y\rangle|^2 \leq 1 - (1/6)^2$ due to the relationship between inner product and trace distance for pure states. Repeating this algorithm a constant number of times yields a bounded-error quantum algorithm for f^x. ◀

3.3 Relation with randomized sabotage complexity

We start by reviewing the definition of randomized sabotage complexity, as presented in [11]. Fix a (partial) Boolean function $f : D \to \{0,1\}$ with $D \in \{0,1\}^n$. For any pair $x, y \in \mathrm{Dom}(f)$ such that $f(x) \neq f(y)$, let $p \in \{0,1,*\}^n$ be the partial assignment of all bits where x and y agree (with the symbol $*$ used for the bits where x and y disagree). We call p a "sabotaged input", imagining that a saboteur replaced bits of x with $*$ symbols until it was no longer possible to determine $f(x)$.

Let $S_* \subseteq \{0,1,*\}^n$ be the set of all sabotaged inputs to f, that is, the set of all partial assignments that are consistent with both a 0-input and a 1-input to f. Let $S_\dagger \in \{0,1,\dagger\}^n$ be the same as S_*, except that the $\dagger$ symbol is used instead of the $*$ symbol. Finally, let $f_{\mathrm{sab}} : S_* \cup S_\dagger \to \{0,1\}$ be the function that takes a sabotaged input and identifies whether it has $*$ symbols or $\dagger$ symbols, promised that it contains only one type of symbol. Intuitively, f_{sab} is a decision problem that forces an algorithm computing it to find a $*$ or $\dagger$. We then define $\mathrm{RS}(f) := \mathrm{R}_0(f_{\mathrm{sab}})$, the expected running time of a zero-error randomized algorithm computing f_{sab}.

To show that $\mathrm{RS}(f)$ is larger than $\mathrm{QD}(f)$ for all f, we will define a classical measure analogous to $\mathrm{QD}(f)$. We will then show this measure is equivalent to $\mathrm{RS}(f)$.

▶ **Definition 9** (Randomized distinguishing complexity)**.** *Let $f : D \to \{0,1\}$, where $D \subseteq \{0,1\}^n$, be an n-bit partial function.* $\mathrm{RD}(f)$ *is defined as the smallest integer k such that there exists a k-query randomized algorithm that on input $x \in D$ outputs a sample from a probability distribution d_x such that*

$$\forall x, y \in D \text{ with } f(x) \neq f(y), \quad D_{\mathrm{TV}}(d_x, d_y) \geq 1/6, \tag{6}$$

where $D_{\mathrm{TV}}(\cdot,\cdot)$ stands for the total variation distance between probability distributions.

Since quantum algorithms can simulate classical algorithms, we immediately get that $\mathrm{QD}(f) \leq \mathrm{RD}(f)$. Next, we will show that $\mathrm{RD}(f) = \Theta(\mathrm{RS}(f))$, completing the argument that $\mathrm{QD}(f) = O(\mathrm{RS}(f))$.

▶ **Theorem 10.** *Let f be a partial Boolean function. Then* $\mathrm{RS}(f)/12 \leq \mathrm{RD}(f) \leq (12/11)\,\mathrm{RS}(f)$.

Proof. First, we show that $\mathrm{RS}(f) \leq 12\,\mathrm{RD}(f)$. Let A be an optimal randomized algorithm for $\mathrm{RD}(f)$, that on input x outputs a sample from the distribution d_x. Let $z \in \mathrm{Dom}(f_{\mathrm{sab}})$ be a sabotaged input, and consider running A on z. Since z is sabotaged, there are inputs x and y with $f(x) \neq f(y)$ that are both consistent with the non-$*$, non-† bits of z. The variation distance between d_x and d_y is at least $1/6$.

A randomized algorithm can be viewed as a probability distribution over deterministic algorithms. Split the support of the distribution for A into two parts: a set S consisting of deterministic algorithms that, when run on z, query a $*$ or †, and a set T consisting of deterministic algorithms that don't query a $*$ or † when run on z. Note that algorithms in T behave the same on x and y. If A samples an algorithm from T with probability p, the total variation distance between the run of A on x and the run of A on y must therefore be at most $2(1-p)$. Since this is at least $1/6$, we have $p \leq 11/12$. Hence when A is run on z, it queries a $*$ or † with probability at least $1/12$.

If we repeat A whenever it does not query a $*$ or †, we get an algorithm that always finds such an entry and uses at most $12\,\mathrm{RD}(f)$ queries on expectation. This is a zero-error randomized algorithm for f_{sab}, so $\mathrm{RS}(f) \leq 12\,\mathrm{RD}(f)$.

We now handle the other direction, showing $\mathrm{RD}(f) \leq (12/11)\,\mathrm{RS}(f)$. Let A be an optimal zero-error randomized algorithm for f_{sab}. It makes $\mathrm{RS}(f)$ queries on expectation, and always finds a $*$ or † in any sabotaged input. Consider the algorithm B that, on input $x \in \mathrm{Dom}(f)$, runs A for at most $2\,\mathrm{RS}(f)$ queries and outputs the partial assignment it queried (that is, it outputs all the pairs (i, x_i) that were queried by the algorithm A).

Let x and y be inputs to f with $f(x) \neq f(y)$. Let z be the sabotaged input defined by x and y, that is, $z_i = *$ if $x_i \neq y_i$ and $z_i = x_i = y_i$ otherwise. By Markov's inequality, after $(12/11)\,\mathrm{RS}(f)$ queries, A finds a $*$ with probability at least $1/12$ when it is run on z. This means that when A is run on x, it queries an index i for which $x_i \neq y_i$ with probability at least $1/12$. When this happens, the output of $B(x)$ is not in the support of d_y. This means d_x puts weight at least $1/12$ on symbols not in the support of d_y. Conversely, d_y puts weight at least $1/12$ on symbols not in the support of d_x. The total variation distance between the two distributions is therefore at least $1/6$, meaning B is a valid $\mathrm{RD}(f)$ algorithm. We conclude that $\mathrm{RD}(f) \leq (12/11)\,\mathrm{RS}(f)$. ◀

Combined with $\mathrm{QD}(f) \leq \mathrm{RD}(f)$, this theorem gives us the following corollary.

▶ **Corollary 11.** *For all (partial) Boolean functions f,* $\mathrm{QD}(f) = O(\mathrm{RS}(f))$.

4 Fifth power query relation

In this section we prove a new relationship between zero-error quantum query complexity and quantum distinguishing complexity and bounded-error quantum query complexity, restated below.

▶ **Theorem 1.** *For all total functions $f : \{0,1\}^n \to \{0,1\}$, we have*

$$\mathrm{Q}_0(f) = O(\mathrm{QD}(f)^5 \log \mathrm{QD}(f)) = O(\mathrm{Q}(f)^5 \log \mathrm{Q}(f)). \tag{1}$$

Additionally, the algorithm also outputs a certificate for $f(x)$ when it outputs $f(x)$.

Our proof uses ideas from an analogous classical result [27, 22] and the main quantum ingredient used is the hybrid argument of Bennett, Bernstein, Brassard, and Vazirani [12]. We now describe and prove a version of the hybrid argument that we use.

4.1 Hybrid argument

We start by defining the concept of a *sensitive block*. For a string $x \in \{0,1\}^n$ and a subset of input bits $B \subseteq [n]$, which we call a block, we use x^B to denote the input with all bits in B flipped. In other words, x^B agrees with x on all positions outside B and disagrees on B. For a function f and an input $x \in \mathrm{Dom}(f)$, we say a block B is a sensitive block if $f(x) \neq f(x^B)$.

Now any algorithm that computes f must also be able to distinguish x from x^B, where B is a sensitive block. Any algorithm that can distinguish x from x^B must "look at" the bits in B in some informal sense. For classical algorithms, this simply means the algorithm has to query a bit from B with high probability. The analogous statement for quantum algorithms is not so clear, since quantum algorithms can query all input bits in superposition. Nevertheless, the hybrid argument still allows us to formalize this intuition in the quantum setting. The hybrid argument asserts that the total weight of queries within the sensitive block (i.e., the total sum of probabilities of querying within the sensitive block over the course of the algorithm) cannot be too small [12]:

▶ **Lemma 12** (Hybrid Argument)**.** *Let $x \in \{0,1\}^n$ be an input, and let $B \subseteq [n]$ be a block. Let Q be a T-query quantum algorithm that accepts x and rejects x^B with high probability, or more generally produces output states that are a constant distance apart in trace distance for x and x^B. Let m_i^t be the probability that, when Q is run on x for t queries and then subsequently measured, it is found to be querying position i of x (i.e., the query register collapses to $|i\rangle$). Then*

$$\sum_{t=1}^{T} \sum_{i \in B} m_i^t = \Omega\left(\frac{1}{T}\right). \tag{7}$$

Note that for a randomized algorithm, we would have $\Omega(1)$ on the right-hand side instead of $\Omega(1/T)$, since a randomized algorithm must look within B (with high probability) at some point during its execution. This lemma was implicitly proven in [12]. We reproduce the proof in Appendix A for the reader's convenience.

4.2 New upper bound

To prove our result we also need to upper bound the number of minimal sensitive blocks of a function. It is not too hard to show that any minimal sensitive block has size at most the sensitivity of f, $s(f)$, which is the maximum number of sensitive blocks of size 1 over all

inputs x. Since there are at most $\binom{n}{\mathrm{s}(f)} = O(n^{\mathrm{s}(f)})$ different subsets of n positions of size $\mathrm{s}(f)$, we know that the number of minimal sensitive blocks is at most this quantity. Kulkarni and Tal [22] improve this simple upper bound replacing n with randomized certificate complexity $\mathrm{RC}(f)$ (Definition 7).

▶ **Lemma 13.** *For any total function $f : \{0,1\}^n \to \{0,1\}$ and any input $x \in \{0,1\}^n$, the number of minimal sensitive blocks of x with respect to f is at most $O(\mathrm{RC}(f)^{\mathrm{s}(f)})$.*

We are now ready to prove Theorem 1.

Proof of Theorem 1. Let Q be the optimal quantum distinguishing algorithm for f, that uses $T = \mathrm{QD}(f)$ queries. Consider running the following quantum algorithm P on oracle input $x \in \{0,1\}^n$:

1. Pick $t \in [T]$ uniformly at random.
2. Run Q on x for t queries and measure the query register.
3. Write down (on a classical tape) the position i where Q is found to be querying, as well as the query output x_i.

The algorithm P uses $t \leq T$ quantum queries. Now that the probability P wrote down the index i is $(1/T)\sum_{t=1}^{T} m_i^t$. For any block $B \subseteq [n]$, the probability that P wrote down some index in B is

$$\frac{1}{T}\sum_{t=1}^{T}\sum_{i\in B} m_i^t. \tag{8}$$

If B is a sensitive block for the input x, then the hybrid argument (Lemma 12) implies the probability that our new algorithm P outputs an index in B is $\Omega(1/T^2)$.

Next, we repeat the algorithm P several times. We claim that after $O(T^2\,\mathrm{s}(f)\log \mathrm{RC}(f))$ repetitions, the outputs of P constitute a certificate for x with constant probability.

To see this, note that for any minimal sensitive block B of the input x, the probability that some run of P (out of the $O(T^2\,\mathrm{s}(f)\log \mathrm{RC}(f))$ many runs) queries in the block B is $1 - O(\mathrm{RC}(f)^{-\mathrm{s}(f)})$. This is because T^2 repetitions boost the probability of querying in a minimal sensitive block from $\Omega(1/T^2)$ to $\Omega(1)$, and then $\mathrm{s}(f)\log \mathrm{RC}(f)$ repetitions of this boosted algorithm further boost the probability to the claimed bound. Hence, by Lemma 13 and the union bound, there is a constant probability that these runs of P query a bit in every minimal sensitive block of the input x. But a set of bits that intersects every sensitive block of x is a certificate for x. Thus these runs of P output a certificate for the input x with constant probability.

Any algorithm that finds a certificate with constant probability can be turned into a zero-error algorithm by repeating whenever a certificate is not found. We therefore get a zero-error algorithm that works simply by repeating P a sufficient number of times. Note that P uses $O(T)$ quantum queries and must be repeated $O(T^2\,\mathrm{s}(f)\log \mathrm{RC}(f))$ times. Recalling that $T = \mathrm{QD}(f)$, we get

$$\mathrm{Q}_0(f) = O(\mathrm{QD}(f)^3\,\mathrm{s}(f)\log \mathrm{RC}(f)). \tag{9}$$

We can simplify this to $\mathrm{Q}_0(f) = O(\mathrm{QD}(f)^5 \log \mathrm{QD}(f))$, since $\mathrm{s}(f) = O(\mathrm{RC}(f)) = O(\mathrm{QC}(f)^2)$ [1] and $\mathrm{QC}(f) = O(\mathrm{QD}(f))$ (Proposition 8). ◀

5 Quantum statistical zero knowledge

5.1 History

The subject of statistical zero-knowledge proof systems has a rich history in the classical setting, and the interested reader is referred to the paper of Sahai and Vadhan [30]. Informally, the complexity class SZK contains problems that can be solved by a probabilistic polynomial-time verifier interacting with a computationally unbounded prover (like the class IP) with the additional restriction that the verifier not learn anything from the prover (statistically) other than the answer to the problem. From this it is clear that BPP $\subseteq$ SZK, since the verifier can simply not interact with the prover, and SZK $\subseteq$ IP, since IP is simply SZK without the zero-knowledge constraint.

More surprisingly, it is also known that SZK = coSZK, and that we can assume without loss of generality that the interaction is only one round and uses public randomness, which means SZK $\subseteq$ AM $\cap$ coAM. Another interesting subtlety is that SZK can be defined assuming an honest verifier, one who does not deviate from the protocol to learn more, or a cheating verifier, who may deviate from the protocol. It turns out that these definitions lead to the same complexity class [17]. The class SZK also has a much simpler characterization in terms of a complete problem called *statistical difference*, as shown by Sahai and Vadhan [30], which yields easier proofs of some of these facts. Informally, in the statistical difference problem we are given two circuits that sample from probability distributions, and the task is determine whether the distributions are far or close in total variation distance.

On the quantum side, (honest-verifier) QSZK was first defined by Watrous [34], and like the classical case, it satisfies BQP $\subseteq$ QSZK $\subseteq$ QIP. The same paper strengthened these obvious containments by showing that QSZK is closed under complement (i.e., QSZK = coQSZK) and that the protocol can be assumed to be one round, which gives QSZK $\subseteq$ QIP(2). Watrous also showed that QSZK has a complete problem, called *quantum state distinguishability*, which is a quantum generalization of the statistical difference problem of Sahai and Vadhan. In this problem, we are given two quantum circuits outputting mixed states and have to decide if the states are far apart or close in trace distance. Later, Watrous [35] also showed that honest-verifier QSZK and cheating-verifier QSZK are the same, as in the classical case.

5.2 Definition

We now define a query analogue of quantum statistical zero-knowledge. Instead of defining QSZK(f) in terms of an interactive zero-knowledge protocol for f, we use the complete problem characterization by Watrous. This yields a considerably simpler definition of QSZK in the query setting.[2]

▶ **Definition 14** (QSZK). *Let $f : D \to \{0,1\}$, where $D \subseteq \{0,1\}^n$, be an n-bit partial function.* QSZK(f) *is defined as the smallest integer k such that there exists two quantum query algorithms making k queries in total that on input $x \in D$ output states ρ_x and σ_x of the same size such that*

- $\forall x \in D$ *with* $f(x) = 1$, $\|\rho_x - \sigma_x\|_{\mathrm{tr}} \geq 2/3$,
- $\forall x \in D$ *with* $f(x) = 0$, $\|\rho_x - \sigma_x\|_{\mathrm{tr}} \leq 1/3$.

[2] The complete problem is often used to define SZK (and its variants, like NISZK) in query complexity and communication complexity (for example, see [13, 33]). It is not obvious whether the definition via an interactive proof and the definition via the complete problem coincide exactly as the problem is complete under polynomial-time reductions, which may add polynomial overhead.

This definition is also robust to some changes. In particular, the constants $2/3$ and $1/3$ can be replaced by any constants $\alpha \in [0,1]$ and $\beta \in [0,1]$ as long as $\alpha^2 > \beta$. Hence an alternate definition with 0.99 instead of $2/3$ and 0.01 instead of $1/3$ leads to the same complexity measure up to multiplicative constants. This follows from the analogous property of the complexity class QSZK, which was shown by Watrous [34] (see Theorem 1 in the conference version or Theorem 5 in the full version for more details).

5.3 Properties

As a sanity check, let us prove the query analog of the obvious containment BQP $\subseteq$ QSZK.

▶ **Proposition 15.** *For all (partial) Boolean functions f, $\mathrm{QSZK}(f) \leq \mathrm{Q}(f)$.*

Proof. Let $\mathrm{Q}(f) = k$ and consider the k-query algorithm that witnesses this fact. Let p_x be the probability that this k-query algorithm when run on input x outputs 1 upon measuring the first qubit. Since the algorithm computes f with bounded error, we know that $p_x \geq 2/3$ for 1-inputs and $p_x \leq 1/3$ for 0-inputs.

Now consider the single-qubit state ρ_x, which is obtained by taking the final state of this algorithm, tracing out all the qubits except the first one, and then applying a completely dephasing channel to it. This is equivalent to measuring the first qubit in the standard basis and outputting $|b\rangle$ when the result is b. This state is $\rho_x = \left(\begin{smallmatrix} 1-p_x & 0 \\ 0 & p_x \end{smallmatrix}\right)$. Let us also define σ_x as $\left(\begin{smallmatrix} 1 & 0 \\ 0 & 0 \end{smallmatrix}\right)$ for all x.

Now let us check that the conditions of Definition 14 are satisfied by these states. For all inputs x, we have $\|\rho_x - \sigma_x\|_{\mathrm{tr}} = |p_x|$. And we know that $p_x \geq 2/3$ for 1-inputs and $0 \leq p_x \leq 1/3$ for 0-inputs, which completes the proof. ◀

The measure $\mathrm{QSZK}(f)$ also satisfies another useful property, that $\mathrm{QSZK}(f) = \Theta(\mathrm{QSZK}(\neg f))$. This is the analogue of the result that QSZK = coQSZK [34]. Since we do not use this property, we only provide a sketch of the proof in Appendix B.

5.4 Relation with adversary bound

We have already showed that $\mathrm{QD}(f) \leq \mathrm{Q}(f)$ (Proposition 6) and $\mathrm{QSZK}(f) \leq \mathrm{Q}(f)$ (Proposition 15). We now show that $\mathrm{QD}(f)$ is actually smaller than $\mathrm{QSZK}(f)$.

▶ **Theorem 2.** *For all (partial) Boolean functions f, $\mathrm{QD}(f) \leq \mathrm{QSZK}(f)$.*

Proof. Let $\mathrm{QSZK}(f) = k$ and consider the quantum algorithms that witnesses this fact. We claim that the tensor product of outputs of these algorithms already satisfies the conditions in Definition 5 and hence proves $\mathrm{QD}(f) \leq k$.

To see this, observe that the algorithm outputs the state $\rho_x \otimes \sigma_x$ on input x, which satisfies the conditions of Definition 14. More precisely, this means for any x and y such that $f(x) = 1$ and $f(y) = 0$, we know that $\|\rho_x - \sigma_x\|_{\mathrm{tr}} \geq 2/3$ and $\|\rho_y - \sigma_y\|_{\mathrm{tr}} \leq 1/3$. We want to show that

$$\|\rho_x \otimes \sigma_x - \rho_y \otimes \sigma_y\|_{\mathrm{tr}} \geq 1/6. \tag{10}$$

Since trace distance is non-increasing under partial trace, we have $\|\rho_x \otimes \sigma_x - \rho_y \otimes \sigma_y\|_{\mathrm{tr}} \geq \|\rho_x - \rho_y\|_{\mathrm{tr}}$ and $\|\rho_x \otimes \sigma_x - \rho_y \otimes \sigma_y\|_{\mathrm{tr}} \geq \|\sigma_x - \sigma_y\|_{\mathrm{tr}}$, which imply

$$\|\rho_x \otimes \sigma_x - \rho_y \otimes \sigma_y\|_{\mathrm{tr}} \geq \max\left\{\|\rho_x - \rho_y\|_{\mathrm{tr}}, \|\sigma_x - \sigma_y\|_{\mathrm{tr}}\right\}.$$

Now if we can show the right-hand side is at least $1/6$, then we are done. To show this, toward a contradiction assume that $\max\left\{\|\rho_x - \rho_y\|_{\mathrm{tr}}, \|\sigma_x - \sigma_y\|_{\mathrm{tr}}\right\} < 1/6$. Then we have

$$\begin{aligned}\|\rho_x - \sigma_x\|_{\mathrm{tr}} &= \|\rho_x - \rho_y + \rho_y - \sigma_y + \sigma_y - \sigma_x\|_{\mathrm{tr}} \\ &\leq \|\rho_x - \rho_y\|_{\mathrm{tr}} + \|\rho_y - \sigma_y\|_{\mathrm{tr}} + \|\sigma_y - \sigma_x\|_{\mathrm{tr}} \\ &< 1/6 + 1/3 + 1/6 = 2/3,\end{aligned}$$

which contradicts $\|\rho_x - \sigma_x\|_{\mathrm{tr}} \geq 2/3$. ◀

As noted, as a corollary of this theorem and $\mathrm{QD}(f) = \Omega(\mathrm{Adv}(f))$, we have for all (partial) functions f, $\mathrm{QSZK}(f) = \Omega(\mathrm{Adv}(f))$.

This can be used to prove lower bounds on QSZK protocols for functions. For example, consider the OR function and let us try to compute it with an interactive protocol without the zero-knowledge requirement. It is easy to see that when $\mathrm{OR}(x) = 1$, a computationally unbounded prover can simply send over the location of a bit i such that $x_i = 1$, which can be checked using only 1 query. Of course, this protocol leaks information and in particular lets the verifier know the location of a 1. But is it necessary that an efficient protocol for OR must leak information? Our lower bound says this must be the case, because $\mathrm{Adv}(\mathrm{OR}) = \Omega(\sqrt{n})$ and hence any zero-knowledge protocol for the function must make $\Omega(\sqrt{n})$ queries.

6 Comparison with other lower bounds

In this section, we establish the separations between quantum distinguishing complexity and the adversary bound and the polynomial method claimed in Theorem 3.

To prove this, we will compose known functions with the index function and establish the behavior of quantum distinguishing complexity under composition with the index function. This kind of composition was also studied by Chen [16], who used it to show an oracle separation between $\mathsf{P}^{\mathsf{SZK}}$ and QSZK.

6.1 Index functions

Let $\textsc{Ind}_k : \{0,1\}^{k+2^k} \to \{0,1\}$ denote the index function, the function that on input (x, y) with $x \in \{0,1\}^k$ and $y \in \{0,1\}^{2^k}$, outputs the bit of y indexed by the string x. We wish to study the composition of the index function with an arbitrary Boolean function f, but composed only on the first k bits of the index function. We'll denote this composition by $\textsc{Ind}_k \circ_k f$. More precisely, if f is an n-bit function, $\textsc{Ind}_k \circ_k f$ is a function on $nk + 2^k$ bits that evaluates f on the first k n-bit strings to obtain a binary string x of length k, and then uses x to index into the next 2^k bits of the input and outputs the bit indexed by x.

In addition to the index function, which is total, we will also study a function we call the "unambiguous index function," $\mathrm{U}\textsc{Ind}_k$. This is a partial function defined similarly to the index function, except that the location of the array y pointed to by the first part of the input is "marked," and we are promised that no other bits of the array are "marked." More explicitly, the function is defined on $k + 2 \cdot 2^k$ bits, with the first k bits indexing a pair of adjacent bits in the remainder of the input. So if the first part of the input represents the integer x, that means it points to the cells $2x$ and $2x + 1$ in the second part of the input. The output of $\mathrm{U}\textsc{Ind}_k$ is the first bit of the pair pointed to, i.e., it will be the bit stored at array location $2x$. Moreover, we are promised that the second bit of this pair (the bit at array location $2x + 1$) will always be 1, and also that the second bit in every *other* pair (i.e., other than the pair $2x$, $2x + 1$) will always be 0.

Intuitively, there is only one strategy to solve $\textsc{Ind}_k$, which is to read the first k bits and find the cell pointed to. But to solve $\textsc{UInd}_k$, there are two good strategies: either read the first k bits (and determine x), or search the remainder of the input for the unique position where the second bit of a pair is 1, which marks the cell pointed to by x.

6.2 Index function composition

We now examine the behavior of quantum distinguishing complexity under composition with the Index and Unambiguous Index functions. To prove our result, we need the following strong direct product theorem for quantum query complexity due to Lee and Roland [25]:

▶ **Theorem 16** (Strong direct product). *Let f be a partial Boolean function with $\mathrm{Dom}(f) \subseteq \{0,1\}^n$, and let $f^{(k)} : \mathrm{Dom}(f)^k \to \{0,1\}^k$ be the task of solving k independent inputs to f simultaneously. Then any quantum algorithm that solves $f^{(k)}$ with success probability at least $(5/6)^k$ uses $\Omega(k\,\mathrm{Q}(f))$ queries.*

We prove the following composition theorem in Appendix C.

▶ **Theorem 17.** *There is a $c > 0$ such that for any partial function f, if $k \geq c \log \mathrm{Q}(f)$, then*

$$\mathrm{QD}(\textsc{Ind}_k \circ_k f) = \Theta(\mathrm{Q}(\textsc{Ind}_k \circ_k f)) = \Theta(k\,\mathrm{Q}(f)) \tag{11}$$

$$\mathrm{QD}(\textsc{UInd}_k \circ_k f) = \Theta(\mathrm{Q}(\textsc{UInd}_k \circ_k f)) = \Theta(k\,\mathrm{Q}(f)). \tag{12}$$

In other words, composing a function with a large index gadget makes QD *and* Q *coincide.*

6.3 Separations

Using this theorem we can now establish Theorem 3, restated for convenience:

▶ **Theorem 3.** *There exist total functions f and g with $\mathrm{QD}(f) = \widetilde{\Omega}(\mathrm{Adv}(f)^2)$ and $\mathrm{QD}(g) \geq \widetilde{\deg}(g)^{4-o(1)}$.*
There also exists an n-bit partial function h with $\mathrm{QD}(h) = \widetilde{\Omega}(n^{1/3})$ and $\mathrm{Adv}(h) = O(\log n)$.

Proof. There exists an n-bit total function f' with a quadratic separation between quantum query complexity and the adversary bound, i.e., $\mathrm{Q}(f') = \widetilde{\Omega}(\mathrm{Adv}(f')^2)$. The function is k-sum with $k \approx \log n$ (see [10, 2] for more details). Now consider the function $f = \textsc{Ind}_k \circ f'$, where $k = \Omega(\log \mathrm{Q}(f))$. By Theorem 17, the QD of these functions increases to Q. However, since the adversary bound satisfies a composition theorem [21], its value only increases by a factor of k. Thus $\mathrm{QD}(f) = \widetilde{\Omega}(\mathrm{Adv}(f)^2)$.

Similarly, if we start with the collision problem which has $\mathrm{Q}(h') = \Theta(n^{1/3})$ [3], but $\mathrm{Adv}(h') = O(1)$, and define $h = \textsc{Ind}_k \circ h'$ for $k = \Theta(\log n)$, then $\mathrm{QD}(h) = \widetilde{\Omega}(n^{1/3})$ but $\mathrm{Adv}(f) = O(\log n)$.

There also exist total functions with $\mathrm{Q}(g') \geq \widetilde{\deg}(g')^{4-o(1)}$ [2]. Composing this function with $\textsc{Ind}_k$ on the first k bits with $k = \Omega(\log \mathrm{Q}(f))$ yields a function g with the desired separation, since approximate polynomial degree also composes in the upper bound direction [31]. ◀

7 Lifting theorems

7.1 Background

Lifting theorems are results that relate communication complexity measures to query complexity measures. For a fixed query measure, such as $\mathrm{D}(f)$, and a communication complexity measure that intuitively corresponds to it, such as deterministic communication complexity

$\mathrm{D}^{\mathrm{cc}}(F)$, we may hope to be able to prove a theorem of the form: there exists some communication gadget G such that $\mathrm{D}^{\mathrm{cc}}(f \circ G) = \widetilde{\Theta}(\mathrm{D}(f))$. In fact, when the size of the gadget G is allowed to depend on the input size of f and the $\widetilde{\Theta}$ is allowed to hide $\operatorname{polylog} n$ factors, such a result is known [19].

We remark that the upper bound direction – showing the communication measure of $f \circ G$ is at most the corresponding query measure of f – is usually easy. We can simulate the query algorithm in the communication complexity world, losing only a multiplicative factor that depends on the difficulty of computing G. The lower bound direction, which lower bounds a communication complexity measure by a query complexity measure, is usually much harder, and is what we will usually refer to when we use the term "lifting theorem."

The result of [19] gives a lifting theorem for deterministic protocols, which we will denote by $\mathrm{D} \to \mathrm{D}^{\mathrm{cc}}$ to mean it transfers a lower bound on the first measure to a lower bound on the second. Recently, lifting theorems have been shown for R and R_0 (with the corresponding communication complexity measures being the obvious ones: randomized communication with bounded error and randomized communication with zero error, denoted R^{cc} and $\mathrm{R}_0^{\mathrm{cc}}$) [20]. We do not know how to lift Q or Q_0 to their analogous communication measures; this is likely to be significantly harder.

7.2 Lifting theorem reductions

In this section, we prove several lifting theorem reductions, showing that a lifting theorem for one measure (such as Q_0) implies a lifting theorem for another measure (such as Q). Our work (including prior work [11]) is the first instance we know of where such reductions are shown; it is perhaps surprising that these reductions can be proven without proving the lifting theorems themselves.

▶ **Theorem 18.** *If there is a lifting theorem for* Q_0 *with gadget* G*, then there is also a lifting theorem for* Q *with the same gadget* G*.*

Proof. Fix a partial function f. We wish to show that $\mathrm{Q}^{\mathrm{cc}}(f \circ G) = \widetilde{\Omega}(\mathrm{Q}(f))$ using a lifting theorem for Q_0.

Let $g = \textsc{UInd}_k \circ_k f$, with $k = \Theta(\mathrm{Q}(f))$. By Theorem 17, we have $\mathrm{Q}(g) = \widetilde{\Omega}(\mathrm{Q}(f))$. Next, apply the lifting theorem to g to get

$$\mathrm{Q}_0^{\mathrm{cc}}(g \circ G) = \widetilde{\Omega}(\mathrm{Q}_0(g)) = \widetilde{\Omega}(\mathrm{Q}(g)) = \widetilde{\Omega}(\mathrm{Q}(f)). \tag{13}$$

To complete the argument, it remains to show that $\mathrm{Q}_0^{\mathrm{cc}}(g \circ G)) = \widetilde{O}(\mathrm{Q}^{\mathrm{cc}}(f \circ G))$. Note that $g \circ G = \textsc{UInd}_k \circ_k f \circ G$. If we have a communication protocol for $f \circ G$, we can simulate it k times (and use error reduction) to obtain the correct index with constant error. We can then use the promise of UInd to check if the index is correct, by verifying that the second bit of the pair at that index is 1. This turns the algorithm into a zero-error algorithm. Since $k = O(\log \mathrm{Q}(f))$, our algorithm uses only $\widetilde{O}(\mathrm{Q}^{\mathrm{cc}}(f \circ G))$ communication. Thus $\mathrm{Q}^{\mathrm{cc}}(f \circ G) = \widetilde{\Omega}(\mathrm{Q}(f))$, as desired. ◀

▶ **Theorem 19.** *If there is a lifting theorem for* QSZK *with gadget* G*, then there is also a lifting theorem for* Q *with the same gadget* G*.*

By a lifting theorem for QSZK, we mean a theorem that lifts it to some communication complexity analogue $\mathrm{QSZK}^{\mathrm{cc}}$. The only property we use of $\mathrm{QSZK}^{\mathrm{cc}}$ is that it lower bounds Q^{cc}.

Proof. Let f be a partial function. Let $g = \textsc{Ind}_k \circ_k f$, where $k = \Theta(\log \mathrm{Q}(f))$. By Theorem 17, $\mathrm{QD}(g) = \widetilde{\Omega}(\mathrm{Q}(f))$. By Theorem 2, $\mathrm{QSZK}(g) = \Omega(\mathrm{QD}(g)) = \widetilde{\Omega}(\mathrm{Q}(f))$. Then

$$\mathrm{Q}^{\mathrm{cc}}(g \circ G) = \Omega(\mathrm{QSZK}^{\mathrm{cc}}(g \circ G)) = \widetilde{\Omega}(\mathrm{QSZK}(g)) = \widetilde{\Omega}(\mathrm{Q}(f)). \tag{14}$$

Also, note that if we had a quantum communication protocol for $f \circ G$ we could easily convert it to a communication protocol for $g \circ G = \textsc{Ind}_k \circ_k f \circ G$. Thus $\mathrm{Q}^{\mathrm{cc}}(f \circ G) = \widetilde{\Omega}(\mathrm{Q}^{\mathrm{cc}}(g \circ G) = \widetilde{\Omega}(\mathrm{Q}(f))$, as desired. ◀

▶ **Theorem 20.** *If there is a lifting theorem that lifts* $\mathrm{QD} \to \mathrm{Q}^{\mathrm{cc}}$ *with gadget* G*, then there is also a lifting theorem for* Q *with the same gadget* G*.*

By a lifting theorem for $\mathrm{QD} \to \mathrm{Q}^{\mathrm{cc}}$, we mean a theorem that shows $\mathrm{Q}^{\mathrm{cc}}(f \circ G) = \widetilde{\Omega}(\mathrm{QD}(f))$ for all partial functions f. This is formally easier to prove than a $\widetilde{\Omega}(\mathrm{Q}(f))$ lower bound, but we show it is actually equivalent.

Proof. Let f be a partial function. Let $g = \textsc{Ind}_k \circ_k f$, where $k = \Theta(\log \mathrm{Q}(f))$. By Theorem 17, $\mathrm{QD}(g) = \widetilde{\Omega}(\mathrm{Q}(f))$. Then

$$\mathrm{Q}^{\mathrm{cc}}(g \circ G) = \widetilde{\Omega}(\mathrm{QD}(g)) = \widetilde{\Omega}(\mathrm{Q}(f)). \tag{15}$$

Also, note that if we had a quantum communication protocol for $f \circ G$ we could easily convert it to a communication protocol for $g \circ G = \textsc{Ind}_k \circ_k f \circ G$. Thus $\mathrm{Q}^{\mathrm{cc}}(f \circ G) = \widetilde{\Omega}(\mathrm{Q}^{\mathrm{cc}}(g \circ G) = \widetilde{\Omega}(\mathrm{Q}(f))$, as desired. ◀

In summary, what we have shown is that a lifting theorem for Q is implied by a lifting theorem for either Q_0, QSZK, or a $\mathrm{QD} \to \mathrm{Q}^{\mathrm{cc}}$ lifting theorem. In fact, each of these statements also has a classical analogue which remains true. Proving a lifting theorem for R_0, SZK, or $\mathrm{RS} \to \mathrm{R}^{\mathrm{cc}}$ would imply a lifting theorem for R. This can be proved analogously; the only property we need is that $\mathrm{RS}(\textsc{UInd}_k \circ_k f) = \widetilde{\Omega}(\mathrm{R}(f))$ when k is at least polylogarithmic in $\mathrm{R}(f)$. An equivalent statement to this was proven in [11]. However, since lifting theorems for R and R_0 are already known (with an index gadget [20]), this reduction is less interesting in the classical case, though it might still be relevant for proving lifting theorems with other gadgets.

References

1. Scott Aaronson. Quantum certificate complexity. *Journal of Computer and System Sciences*, 74(3):313–322, 2008. doi:10.1016/j.jcss.2007.06.020.
2. Scott Aaronson, Shalev Ben-David, and Robin Kothari. Separations in query complexity using cheat sheets. In *Proceedings of the 48th Symposium on Theory of Computing (STOC 2016)*, pages 863–876, 2016. doi:10.1145/2897518.2897644.
3. Scott Aaronson and Yaoyun Shi. Quantum Lower Bounds for the Collision and the Element Distinctness Problems. *Journal of the ACM*, 51(4):595–605, 2004. doi:10.1145/1008731.1008735.
4. Andris Ambainis. Quantum Lower Bounds by Quantum Arguments. *Journal of Computer and System Sciences*, 64(4):750–767, June 2002. doi:10.1006/jcss.2002.1826.
5. Andris Ambainis. Polynomial Degree vs. Quantum Query Complexity. In *Proceedings of the 54th Symposium on Foundations of Computer Science (FOCS 2003)*, page 230, 2003. doi:10.1109/SFCS.2003.1238197.
6. Andris Ambainis, Kaspars Balodis, Aleksandrs Belovs, Troy Lee, Miklos Santha, and Juris Smotrovs. Separations in Query Complexity Based on Pointer Functions. In *Proceedings of the 48th Symposium on Theory of Computing*, STOC '16, pages 800–813, 2016. doi:10.1145/2897518.2897524.

7 Howard Barnum, Michael Saks, and Mario Szegedy. Quantum Decision Trees and Semidefinite Programming. Technical report, Los Alamos National Laboratory, 2001. URL: http://permalink.lanl.gov/object/view?what=info:lanl-repo/lareport/LA-UR-01-6417.

8 Howard Barnum, Michael Saks, and Mario Szegedy. Quantum query complexity and semidefinite programming. In *18th Conference on Computational Complexity (CCC 2003)*, pages 179–193, 2003. doi:10.1109/CCC.2003.1214419.

9 Robert Beals, Harry Buhrman, Richard Cleve, Michele Mosca, and Ronald de Wolf. Quantum lower bounds by polynomials. *Journal of the ACM*, 48(4):778–797, 2001. doi:10.1145/502090.502097.

10 Aleksandrs Belovs and Robert Spalek. Adversary lower bound for the k-sum problem. In *Proceedings of the 4th conference on Innovations in Theoretical Computer Science*, pages 323–328, 2013. doi:10.1145/2422436.2422474.

11 Shalev Ben-David and Robin Kothari. Randomized Query Complexity of Sabotaged and Composed Functions. In *43rd International Colloquium on Automata, Languages, and Programming (ICALP 2016)*, volume 55 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 60:1–60:14, 2016. doi:10.4230/LIPIcs.ICALP.2016.60.

12 Charles H. Bennett, Ethan Bernstein, Gilles Brassard, and Umesh Vazirani. Strengths and Weaknesses of Quantum Computing. *SIAM Journal on Computing (special issue on quantum computing)*, 26:1510–1523, 1997. doi:10.1137/S0097539796300933.

13 Adam Bouland, Lijie Chen, Dhiraj Holden, Justin Thaler, and Prashant Nalini Vasudevan. On the Power of Statistical Zero Knowledge. In *58th Annual Symposium on Foundations of Computer Science (FOCS 2017)*, pages 708–719, October 2017. doi:10.1109/FOCS.2017.71.

14 Harry Buhrman, Richard Cleve, Ronald de Wolf, and Christof Zalka. Bounds for Small-Error and Zero-Error Quantum Algorithms. In *Proceedings of the 40th Symposium on Foundations of Computer Science*, FOCS '99, pages 358–368, 1999. doi:10.1109/SFFCS.1999.814607.

15 Harry Buhrman and Ronald de Wolf. Complexity measures and decision tree complexity: a survey. *Theoretical Computer Science*, 288(1):21–43, 2002. doi:10.1016/S0304-3975(01)00144-X.

16 Lijie Chen. A note on oracle separations for BQP. *arXiv preprint*, 2016. arXiv:1605.00619.

17 Oded Goldreich, Amit Sahai, and Salil Vadhan. Honest-verifier Statistical Zero-knowledge Equals General Statistical Zero-knowledge. In *Proceedings of the 30th Symposium on Theory of Computing*, STOC '98, pages 399–408, 1998. doi:10.1145/276698.276852.

18 Mika Göös, Shachar Lovett, Raghu Meka, Thomas Watson, and David Zuckerman. Rectangles Are Nonnegative Juntas. *SIAM Journal on Computing*, 45(5):1835–1869, 2016. doi:10.1137/15M103145X.

19 Mika Göös, Toniann Pitassi, and Thomas Watson. Deterministic Communication vs. Partition Number. In *Proceedings of the 56th Symposium on Foundations of Computer Science (FOCS 2015)*, pages 1077–1088, 2015. doi:10.1109/FOCS.2015.70.

20 Mika Göös, Toniann Pitassi, and Thomas Watson. Query-to-communication lifting for BPP. In *58th Annual Symposium on Foundations of Computer Science (FOCS 2017)*, pages 132–143, October 2017. doi:10.1109/FOCS.2017.21.

21 Peter Høyer, Troy Lee, and Robert Špalek. Negative weights make adversaries stronger. In *Proceedings of the 39th Symposium on Theory of Computing (STOC 2007)*, pages 526–535, 2007. doi:10.1145/1250790.1250867.

22 Raghav Kulkarni and Avishay Tal. On Fractional Block Sensitivity. *Chicago Journal of Theoretical Computer Science*, 2016(8), July 2016. doi:10.4086/cjtcs.2016.008.

23 Sophie Laplante and Frédéric Magniez. Lower bounds for randomized and quantum query complexity using Kolmogorov arguments. In *Proceedings of the 19th Conference on Computational Complexity*, pages 294–304, June 2004. doi:10.1109/CCC.2004.1313852.

24 Troy Lee, Rajat Mittal, Ben W. Reichardt, Robert Špalek, and Mario Szegedy. Quantum query complexity of state conversion. In *Proceedings of the 52nd Symposium on Foundations of Computer Science (FOCS 2011)*, pages 344–353, 2011. doi:10.1109/FOCS.2011.75.

25 Troy Lee and Jérémie Roland. A strong direct product theorem for quantum query complexity. *Computational Complexity*, 22(2):429–462, 2013. doi:10.1007/s00037-013-0066-8.

26 Sanketh Menda and John Watrous. Oracle Separations for Quantum Statistical Zero-Knowledge. *arXiv preprint*, 2018. arXiv:1801.08967.

27 Gatis Midrijanis. On randomized and quantum query complexities. *arXiv preprint*, 2005. arXiv:quant-ph/0501142.

28 Michael A. Nielsen and Isaac L. Chuang. *Quantum Computation and Quantum Information*. Cambridge University Press, 2000.

29 Ran Raz and Pierre McKenzie. Separation of the Monotone NC Hierarchy. *Combinatorica*, 19(3):403–435, 1999. doi:10.1007/s004930050062.

30 Amit Sahai and Salil Vadhan. A Complete Problem for Statistical Zero Knowledge. *Journal of the ACM*, 50(2):196–249, March 2003. doi:10.1145/636865.636868.

31 Alexander A. Sherstov. Making Polynomials Robust to Noise. In *Proceedings of the 44th Symposium on Theory of Computing*, STOC '12, pages 747–758, 2012. doi:10.1145/2213977.2214044.

32 Robert Špalek and Mario Szegedy. All Quantum Adversary Methods are Equivalent. *Theory of Computing*, 2(1):1–18, 2006. doi:10.4086/toc.2006.v002a001.

33 List of Open Problems in Sublinear Algorithms. Problem 77: Frontiers in Structural Communication Complexity. https://sublinear.info/index.php?title=Open_Problems:77, 2017.

34 John Watrous. Limits on the Power of Quantum Statistical Zero-Knowledge. In *Proceedings of the 43rd Symposium on Foundations of Computer Science*, FOCS '02, pages 459–468, 2002. Full version available at https://cs.uwaterloo.ca/~watrous/Papers/. doi:10.1109/SFCS.2002.1181970.

35 John Watrous. Zero-Knowledge against Quantum Attacks. *SIAM Journal on Computing*, 39(1):25–58, 2009. doi:10.1137/060670997.

36 John Watrous. *The Theory of Quantum Information*. Cambridge University Press, 2018. Available at https://cs.uwaterloo.ca/~watrous/TQI/.

37 Shengyu Zhang. On the power of Ambainis lower bounds. *Theoretical Computer Science*, 339(2):241–256, 2005. doi:10.1016/j.tcs.2005.01.019.

A Proof of the hybrid argument

In this section, we prove Lemma 12, restated below for convenience.

▶ **Lemma 12** (Hybrid Argument). *Let $x \in \{0,1\}^n$ be an input, and let $B \subseteq [n]$ be a block. Let Q be a T-query quantum algorithm that accepts x and rejects x^B with high probability, or more generally produces output states that are a constant distance apart in trace distance for x and x^B. Let m_i^t be the probability that, when Q is run on x for t queries and then subsequently measured, it is found to be querying position i of x (i.e., the query register collapses to $|i\rangle$). Then*

$$\sum_{t=1}^{T} \sum_{i \in B} m_i^t = \Omega\left(\frac{1}{T}\right). \tag{7}$$

Proof. We start by fixing some notation. Let the quantum query algorithm Q act on m qubits, initialized in the all-zeros state $|0^m\rangle$. A T-query algorithm is specified by $T+1$ unitaries $U_0, U_1, \ldots, U_T$ acting on m qubits. For any input $x \in \{0,1\}^n$, the oracle O_x acts as $O_x|i,b\rangle = |i, b \oplus x_i\rangle$ for all $i \in [n]$ and $b \in \{0,1\}$. The output state produced by this quantum algorithm (before measurement) on input x is

$$|\psi_x\rangle = U_T O_x U_{T-1} O_x \cdots O_x U_1 O_x U_0 |0^m\rangle, \tag{16}$$

where O_x is implicitly $O_x \otimes \mathbb{1}$ if O_x acts on fewer than m qubits. Within the m qubits, we further group the qubits into three registers, the first register holds an index $|i\rangle$, for $i \in [n]$, the second holds a qubit $|b\rangle$, for $b \in \{0,1\}$, and the third register contains all the remaining qubits.

For a quantum algorithm outputting a Boolean function, we assume that the first qubit of $|\psi_x\rangle$ is measured at the end to determine the output. A quantum distinguishing algorithm may trace out some qubits of $|\psi_x\rangle$ before producing an output or it may simply output the state $|\psi_x\rangle$ without loss of generality, since tracing out qubits cannot increase the distance between a pair of states.

In our case we have an algorithm Q that accepts x and rejects x^B with high probability. To be more concrete, let us assume Q has error probability ϵ. As we saw in Proposition 6, such an algorithm can be made to output a mixed state ρ_x such that $\|\rho_x - \rho_{x^B}\|_{\mathrm{tr}} \geq 1 - 2\epsilon$. Since trace distance is non-increasing under partial trace [28, Th. 9.2], we get that the pure output states must also be far, and hence $\||\psi_x\rangle\langle\psi_x| - |\psi_{x^B}\rangle\langle\psi_{x^B}|\|_{\mathrm{tr}} \geq 1 - 2\epsilon$. This is all we need to assume about the output of the algorithm on these inputs.

We now consider the intermediate states produced by this quantum algorithm after t queries to input x. Let

$$|\psi_x^0\rangle := U_0|0^m\rangle \quad \text{and} \quad |\psi_x^t\rangle := U_t O_x |\psi_x^{t-1}\rangle. \tag{17}$$

for $t \in [T]$. The final state of the algorithm is $|\psi_x^T\rangle = |\psi_x\rangle$, and hence we have

$$\left\||\psi_x^T\rangle\langle\psi_x^T| - |\psi_{x^B}^T\rangle\langle\psi_{x^B}^T|\right\|_{\mathrm{tr}} \geq 1 - 2\epsilon. \tag{18}$$

We know that the states are far apart in trace distance, but we also want to bound their closeness in ℓ_2 distance. By (3), we have

$$|\langle\psi_x^T|\psi_{x^B}^T\rangle| \leq \sqrt{1-(1-2\epsilon)^2} = 2\sqrt{\epsilon(1-\epsilon)} \leq 1 - (1/2)(1-2\epsilon)^2. \tag{19}$$

Then we have

$$\begin{aligned} \||\psi_x^T\rangle - |\psi_{x^B}^T\rangle\|^2 &= 2 - \langle\psi_{x^B}^T|\psi_x^T\rangle - \langle\psi_x^T|\psi_{x^B}^T\rangle \\ &= 2 - 2\,\mathrm{Re}(\langle\psi_{x^B}^T|\psi_x^T\rangle) \geq 2 - 2|\langle\psi_{x^B}^T|\psi_x^T\rangle| \geq (1-2\epsilon)^2, \end{aligned} \tag{20}$$

and so $\||\psi_x^T\rangle - |\psi_{x^B}^T\rangle\| \geq 1 - 2\epsilon$.

Hence the final states of the algorithm are far in apart in ℓ_2 distance on inputs x and x^B. We also know that the initial states $|\psi_x^0\rangle$ and $|\psi_{x^B}^0\rangle$ are identical. We keep track of how much this distance $d_t := \||\psi_x^t\rangle - |\psi_{x^B}^t\rangle\|$ changes for $t \in \{0,1,\ldots,T\}$. For each t, we have

$$d_{t+1} = \||\psi_x^{t+1}\rangle - |\psi_{x^B}^{t+1}\rangle\| = \|U_{t+1}O_x|\psi_x^t\rangle - U_{t+1}O_{x^B}|\psi_{x^B}^t\rangle\| = \|O_x|\psi_x^t\rangle - O_{x^B}|\psi_{x^B}^t\rangle\|, \tag{21}$$

since U_{t+1} is a unitary and preserves norms. This equals

$$\|O_{x^B}|\psi_x^t\rangle - O_{x^B}|\psi_{x^B}^t\rangle + (O_x - O_{x^B})|\psi_x^t\rangle\| \leq \|O_{x^B}|\psi_x^t\rangle - O_{x^B}|\psi_{x^B}^t\rangle\| + \|(O_x - O_{x^B})|\psi_x^t\rangle\| \tag{22}$$

$$= d_t + \|(O_x - O_{x^B})|\psi_x^t\rangle\|. \tag{23}$$

Next, decompose $|\psi_x^t\rangle$ by the value of the query register. On basis vectors when the query register is not in B, the unitaries O_x and O_{x^B} behave the same; such vectors therefore get mapped to zero. If $|\psi_t^{x,B}\rangle$ denotes the component of $|\psi_x^t\rangle$ whose query register is in B, we get

$$\|(O_x - O_{x^B})|\psi_x^t\rangle\| = \|(O_x - O_{x^B})|\psi_t^{x,B}\rangle\| \leq \|O_x|\psi_t^{x,B}\rangle\| + \|O_{x^B}|\psi_t^{x,B}\rangle\| = 2\||\psi_t^{x,B}\rangle\| \tag{24}$$

$$= 2 \cdot \sqrt{\sum_{i \in B} m_i^{t+1}}, \tag{25}$$

where the last equality follows from the definition of m_i^{t+1}, which is defined to be the probability that the algorithm is found to be querying position i right before making query $t+1$. The increase from d_t to d_{t+1} is therefore upper bounded by $2\sqrt{\sum_{i \in B} m_i^{t+1}}$, so we have

$$2 \sum_{t=1}^{T} \sqrt{\sum_{i \in B} m_i^t} \geq d_T - d_0 \geq 1 - 2\epsilon. \tag{26}$$

Using the Cauchy–Schwarz inequality on the outer sum gives

$$2\sqrt{T} \sqrt{\sum_{t=1}^{T} \sum_{i \in B} m_i^t} \geq 1 - 2\epsilon, \tag{27}$$

or

$$\sum_{t=1}^{T} \sum_{i \in B} m_i^t \geq \frac{(1-2\epsilon)^2}{4T} = \Omega\left(\frac{1}{T}\right), \tag{28}$$

when ϵ is a constant.[3] ◀

B QSZK closed under complement

Sketch of proof of $\mathrm{QSZK}(f) = \Theta(\mathrm{QSZK}(\neg f))$. To prove this, we would like to reduce the complete problem to its complement. In other words, we are given two circuits that query an oracle preparing ρ_x and σ_x that are either far apart in trace distance (when $f(x) = 1$) or close in trace distance (when $f(x) = 0$). From these circuits, we want to define two new states ρ'_x and σ'_x, such that these states are far when ρ_x and σ_x were close, and close when ρ_x and σ_x were far. Before starting the transformation, we first boost the parameters $2/3$ and $1/3$ to be extremely close to 1 and 0 respectively. For this sketch we will assume the parameters are exactly 1 and 0, which means when the states are far, they are perfectly distinguishable (i.e., $\|\rho_x - \sigma_x\|_{\mathrm{tr}} = 1$), and when they are close, they are equal (i.e., $\rho_x = \sigma_x$).

To perform this transformation, consider the pure states output by the circuits before tracing out any qubits. Let $|R_x\rangle_{BC}$ and $|S_x\rangle_{BC}$ be the pure state on registers B and C, which yields ρ_x and σ_x, respectively when register B is traced out. More formally, we have

$$\rho_x = \mathrm{Tr}_B(|R_x\rangle\langle R_x|_{BC}) \text{ and } \sigma_x = \mathrm{Tr}_B(|S_x\rangle\langle S_x|_{BC}). \tag{29}$$

From the pure states $|R_x\rangle_{BC}$ and $|S_x\rangle_{BC}$, we define two new pure states on registers A, B, C, and D, as follows:

$$|R'_x\rangle = \frac{1}{\sqrt{2}}\Big(|0\rangle_A |R_x\rangle_{BC} |0\rangle_D + |1\rangle_A |S_x\rangle_{BC} |0\rangle_D\Big) \text{ and} \tag{30}$$

$$|S'_x\rangle = \frac{1}{\sqrt{2}}\Big(|0\rangle_A |R_x\rangle_{BC} |0\rangle_D + |1\rangle_A |S_x\rangle_{BC} |1\rangle_D\Big). \tag{31}$$

[3] This can be slightly improved to $(1 - 2\sqrt{\epsilon(1-\epsilon)})/2T$ by not using the approximation in (19).

Note that the only difference between these states is in register D. If we have circuits preparing states $|R_x\rangle_{BC}$ and $|S_x\rangle_{BC}$, it is easy to see that we can construct circuits preparing $|R'_x\rangle_{ABCD}$ and $|S'_x\rangle_{ABCD}$. We now define the states ρ'_x and σ'_x from these states by tracing out registers C and D:

$$\rho'_x = \mathrm{Tr}_{CD}(|R'_x\rangle\langle R'_x|_{ABCD}) \text{ and } \sigma'_x = \mathrm{Tr}_{CD}(|S'_x\rangle\langle S'_x|_{ABCD}). \tag{32}$$

We claim that these states satisfy the conditions we require. When $f(x) = 1$, we have that $\|\rho_x - \sigma_x\|_{\mathrm{tr}} = 1$, i.e., the residual state on register C for states $|R'_x\rangle$ and $|S'_x\rangle$ is completely distinguishable. In this case, before we trace out registers C and D, we could implement a unitary on these registers which reads register C and writes onto register D whether the state in C is ρ_x or σ_x. This operation maps the state $|R'_x\rangle$ to the state $|S'_x\rangle$ and only acts on the traced out qubits, which does not affect the qubits that are not traced out, and we have $\rho'_x = \sigma'_x$.

When $f(x) = 0$, we have that $\rho_x = \sigma_x$. In this case we want to show that ρ'_x and σ'_x are distinguishable. We will show that after applying a specific unitary to these states are tracing out register B, in the first case we are left with the state $|+\rangle\langle+|_A$, but in the second case we have $\frac{1}{2}\mathbb{1}_A$, which can be distinguished.

Since $\rho_x = \sigma_x$, there is a unitary U_B such that $(U_B \otimes \mathbb{1}_C)|R_x\rangle_{BC} = |S_x\rangle_{BC}$. Controlled on the qubit in register A, let us apply the unitary U_B to register B of $|R'_x\rangle$ and $|S'_x\rangle$ before we trace out registers C and D, which is equivalent to applying it after tracing out the registers. This makes registers BC unentangled with the rest of the state, and equal to $|S_x\rangle_{BC}$. In the first case we are left with the state $|+\rangle_A|0\rangle_D$ on registers A and D, while in the second case we have $\frac{1}{2}(|00\rangle_{AD} + |11\rangle_{AD})$. Tracing out register D leaves us with the $|+\rangle$ state in the first case and the maximally mixed state in the second case, as claimed. ◁

C Proof of Theorem 17

Proof. Recall that quantum query complexity composes perfectly [24], so $\mathrm{Q}(\textsc{Ind}_k \circ f) = \Theta(\mathrm{Q}(\textsc{Ind}_k)\,\mathrm{Q}(f)) = O(k\,\mathrm{Q}(f))$. We argue that $\mathrm{Q}(\textsc{Ind}_k \circ_k f)$ is smaller than $\mathrm{Q}(\textsc{Ind}_k \circ f)$. This is because we can convert any algorithm for $\mathrm{Q}(\textsc{Ind}_k \circ f)$ into an algorithm for $\mathrm{Q}(\textsc{Ind}_k \circ_k f)$: fix a 0-input x^0 and a 1-input x^1 for f; then, given an input to $\mathrm{Q}(\textsc{Ind}_k \circ_k f)$, pretend that each 0 bit in the second half of the input is actually x^0, and that each 1 bit is actually x^1 (the algorithm can do this by applying the appropriate unitary). This converts the input into an input for $\mathrm{Q}(\textsc{Ind}_k \circ f)$, completing the reduction.

Thus $\mathrm{Q}(\textsc{Ind}_k \circ_k f) = O(k\,\mathrm{Q}(f))$. Similarly, $\mathrm{Q}(\textsc{UInd}_k \circ_k f) = O(k\,\mathrm{Q}(f))$. Since QD is smaller than Q, it remains only to show that $\mathrm{QD}(\textsc{Ind}_k \circ_k f) = \Omega(k\,\mathrm{Q}(f))$ and $\mathrm{QD}(\textsc{UInd}_k \circ_k f) = \Omega(k\,\mathrm{Q}(f))$. We complete the argument for UIND; the argument for IND is similar.

Let Q be an optimal quantum distinguishing algorithm for $\textsc{UInd}_k \circ_k f$. We turn Q into a quantum algorithm Q' that uses the same number of queries, and solves all k copies of f with non-negligible probability; we then apply the direct product theorem (Theorem 16) to lower bound the number of queries required by Q', and hence by Q.

Given k inputs to f, the first thing the algorithm Q' does is append an all-0 array to turn it into an input to $\textsc{UInd}_k \circ_k f$. (Since the array is all zeros, the new input does not satisfy the promise of $\textsc{UInd}_k \circ_k f$, but we will still be able to run Q on it.) Then Q' picks a random number t between 1 and T uniformly, where $T = \mathrm{QD}(\textsc{UInd}_k \circ_k f)$ is the number of queries used by Q, and simulates Q for t queries. The algorithm Q' then measures the state of Q to determine the position at which Q was going to query. If this position is in the array part of the input and is inside a pair that has index $i \in \{0,1\}^k$, the algorithm Q' will then output the string i.

Consider the correct pair in the array (the one really pointed to by the k copies of f). Flipping the pair from 00 to 01 causes the input to satisfy the promise of $\text{UIND}_k \circ_k f$, and causes the output to become a 0-input. On the other hand, flipping the pair from 00 to 11 causes the input to become a 1-input. Let $|\psi\rangle$ be the final state of Q when run on the original, illegal input. Let $|\psi_0\rangle$ be the final state of Q when run on the flipped 0-input, and let $|\psi_1\rangle$ be the final state of Q when run on the 1-input. We know that $|\psi_0\rangle$ and $|\psi_1\rangle$ are far in trace distance. Hence $|\psi\rangle$ must be a far in trace distance from at least one on them.

Thus by Lemma 12, the probability that Q' finds Q querying inside the correct pair of the array is $\Omega(1/T^2)$. This means that Q' outputs the correct string of answers to the k inputs to f is with probability at least $\Omega(1/T^2)$. Since Q' uses only T queries, by Theorem 16 we must have either $T = \Omega(k\,\mathrm{Q}(f))$ or $1/T^2 = O((5/6)^k)$. The latter implies $T = \Omega((6/5)^{k/2}) = \Omega((6/5)^{k/4} \cdot (6/5)^{k/4}) = 2^{\Omega(k)} \cdot 2^{\Omega(k)}$. When $k \geq c \log \mathrm{Q}(f)$ for a large enough constant c, this gives $T \geq 2^{\Omega(k)}\,\mathrm{Q}(f) = \Omega(k\,\mathrm{Q}(f))$. Recalling that $T = \mathrm{QD}(\text{UIND}_k \circ_k f)$, we get $\mathrm{QD}(\text{IND}_k \circ_k f) = \Omega(k\,\mathrm{Q}(f))$, as desired. ◀

Circuit Transformations for Quantum Architectures

Andrew M. Childs
Joint Center for Quantum Information and Computer Science, University of Maryland, USA
Institute for Advanced Computer Studies, University of Maryland, USA
Department for Computer Science, University of Maryland, USA
amchilds@umd.edu

Eddie Schoute
Joint Center for Quantum Information and Computer Science, University of Maryland, USA
Institute for Advanced Computer Studies, University of Maryland, USA
Department for Computer Science, University of Maryland, USA
eschoute@cs.umd.edu

Cem M. Unsal
Department of Mathematics, University of Maryland, USA

Abstract

Quantum computer architectures impose restrictions on qubit interactions. We propose efficient circuit transformations that modify a given quantum circuit to fit an architecture, allowing for any initial and final mapping of circuit qubits to architecture qubits. To achieve this, we first consider the qubit movement subproblem and use the Routing via Matchings framework to prove tighter bounds on parallel routing. In practice, we only need to perform partial permutations, so we generalize Routing via Matchings to that setting. We give new routing procedures for common architecture graphs and for the generalized hierarchical product of graphs, which produces subgraphs of the Cartesian product. Secondly, for serial routing, we consider the Token Swapping framework and extend a 4-approximation algorithm for general graphs to support partial permutations. We apply these routing procedures to give several circuit transformations, using various heuristic qubit placement subroutines. We implement these transformations in software and compare their performance for large quantum circuits on grid and modular architectures, identifying strategies that work well in practice.

2012 ACM Subject Classification Computer systems organization → Quantum computing; Hardware → Quantum computation; Mathematics of computing → Graph theory; Applied computing → Physics; General and reference → General conference proceedings; Networks

Keywords and phrases quantum circuit, quantum architectures, circuit mapping

Digital Object Identifier 10.4230/LIPIcs.TQC.2019.3

Related Version Full version at `arXiv:1902.09102 [quant-ph]`.

Supplement Material Source code and result data available at `https://gitlab.umiacs.umd.edu/amchilds/arct`.

Funding This work was supported in part by the Army Research Office (MURI award number W911NF-16-1-0349), the Canadian Institute for Advanced Research, the National Science Foundation (grant number 1813814), and the U.S. Department of Energy, Office of Science, Office of Advanced Scientific Computing Research, Quantum Algorithms Teams and Quantum Testbed Pathfinder (award number DE-SC0019040) programs.

Acknowledgements The authors would like to thank Aniruddha Bapat for insights on the hierarchical product of graphs and suggestions for tightening the routing lower bound for these graphs. We would also like to thank Drew Risinger for helpful formative discussions.

1 Introduction

Quantum algorithms are typically formulated in a circuit model in which two-qubit gates can be performed between any pair of qubits. However, most realistic quantum architectures impose restrictions on qubit interactions. Thus a natural challenge is to find a way of implementing a given circuit on a given architecture with low overhead. We can do this by finding a time-efficient *architecture-respecting circuit transformation* – a mapping to a new circuit that preserves the function of the original quantum circuit up to an initial mapping of circuit qubits to architecture qubits and a final mapping of architecture qubits back to circuit qubits, where the new circuit is constrained to respect the architecture.

There have been many proposals for the design of quantum processors. Examples include trapped ion systems that enable interactions between any two ions in a trap [39] and superconducting qubit architectures with more limited interactions [19, 24, 44]. Many proposed architectures for scalable devices employ modularity, building a large device from interconnected subunits [39, 40, 12].

There is also a considerable amount of work on implementing circuits under architectural constraints. Some examples include implementations of Shor's algorithm [18], the quantum Fourier transform on 1D nearest-neighbor architectures [35], and quantum adders on nearest-neighbor architectures [15, 16]. However, the aforementioned works focus on analyzing specific circuits. Instead, we wish to find automated circuit transformations that can handle complex circuits and compare their performance when implemented in various architectures. Bounds on the efficacy of architecture-respecting circuit transformations and good automated tools for implementing them may be able to inform architecture design decisions [52]. Unfortunately, it is challenging to achieve good performance with an automated tool. Indeed, finding even one optimal placement for a set of gates is NP-hard [34].

Prior Work on Automated Architecture-Respecting Circuit Transformations

Several previous works use exhaustive approaches that take time exponential in the number of qubits (and hence can only be used for small instances). For example, Saeedi, Wille, and Drechsler [46] use SAT solvers to decompose circuits so they can be run on the path architecture; [33] finds an optimal circuit transformation on nearest-neighbor architectures by formulating the problem as a pseudo-boolean optimization; Venturelli et al. [50] use temporal planners to schedule gates; and [41] uses satisfiability modulo theory solvers to find mappings of the circuit with high success probability using calibration data. Other work has instead proposed minimizing the distance between all qubits in groups of gates on specific architectures [48, 54, 43], but this is also NP-hard in general. These and other papers add SWAP gates so that the logical state of a given physical qubit is transferred to a different physical qubit (henceforth, we simply refer to this as *qubit movement*, with the implicit understanding that only the logical state is moved).

As a heuristic solution, we can break the circuit into sets of disjoint gates and move qubits between each set. Metodi et al. [36] propose polynomial-time heuristic routines that prioritize gates with many dependents. Hirata et al. [22] propose exhaustive and heuristic searches for good placements of qubits on the path architecture to construct circuit transformations.

One can also use heuristic qubit placement and movement algorithms on fault-tolerant 2D grid architectures [30] or algorithms that are designed to handle the surface code [28]. We do not consider fault tolerance explicitly and instead work only at the logical level.

An exhaustive search of all permutations of n qubit locations takes time $O(n!)$ but can work well for small numbers of locations [53], or can be done selectively using A^* heuristic search [57, 58] or local search [34, 8]. By choosing a suitable initial placement of qubits, we

can further reduce the qubit movement cost. For example, [29] tries to find a good initial placement by repeatedly transforming the quantum circuit forwards and then backwards, taking the output qubit placement as input for the next iteration.

Others have considered a model in which one can perform fast measurements and adapt later parts of the computation based on the outcomes [20]. This model allows the movement of qubits with just a constant overhead at the cost of extra ancillas [45]. However, realizing such a model presents significant technical challenges and we do not consider it here.

Various bounds are also known for the cost of moving qubits. Sorting networks provide a way to upper bound the depth of the qubit movement circuit [27, 7, 13, 21]. We refer to [3] for a more complete overview of sorting networks.

Contribution

In this paper, we construct architecture-respecting circuit transformations that attempt to minimize the circuit depth or size overhead and have worst-case time complexity polynomial in the sizes of the circuit and architecture graph. We model the connectivity of the underlying hardware as a simple graph where vertices represent the qubits and edges represent places where a two-qubit gate can be performed.

As a simple and fast approach, we propose the *greedy swap circuit transformation* (Section 2.2.2). It inserts SWAPs on edges chosen to minimize the total distance between qubits involved in two-qubit gates until some gate(s) can be executed.

We then propose building architecture-respecting circuit transformations (Section 2.2.3) by combining algorithms for two basic subproblems: qubit movement (addressed by *permuters*, for which we provide theoretical performance guarantees) and qubit placement (addressed by *mappers*). For the latter, we specify a variety of heuristic strategies (Section 4) to find suitable placements of qubits from the input circuit, attempting to optimize for circuit size or depth. We implement these algorithms in software, which is publicly available under a free software license [47].

Consider now the problem of moving qubits on a given architecture graph. A sorting network sorts any fixed-length sequence of integers with a circuit of comparators, which compare two inputs and output them in some ordering. While sorting networks can be used to route qubits [7], they achieve a more general task, and the cost of routing can sometimes be lower with other methods. Specifically, we suggest Routing via Matchings [2] (introduced in Section 3.1) as a more suitable framework for moving qubits in parallel. Deciding whether there exists a depth-k circuit for Routing via Matchings is NP-complete in general for $k > 2$ [3], but optimal or near-optimal protocols are known for specific graph families [2, 56]. In some cases it is possible to implement any permutation asymptotically more efficiently than a general sorting network (see Table 1). On complete graphs, for example, any permutation can be implemented in a depth-2 circuit of transpositions [2], whereas an optimal sorting network has depth $\Theta(\log n)$ [1].

While it is common to consider only the worst-case routing performance, we also wish to route efficiently in practice. To improve practical performance, we generalize to *partial permutations* (permutations only defined on some subdomain) so that we can also move subsets of qubits efficiently. The destinations of the remaining qubits are unconstrained. In Section 3.1, we present routing algorithms for the path graph, the complete graph, and the *generalized hierarchical product* of graphs [6], which includes the Cartesian product of graphs and *modular* architectures as special cases [40]. Graphs obtained as hierarchical products have many good properties for quantum architectures [5]. We establish an upper bound on the routing number of a hierarchical product (Theorem 4) that matches prior work for total permutations on the Cartesian product of graphs [2] and depends on easily computable properties of the input partial permutation.

Table 1 Performance bounds for sorting networks versus routing via matchings (the routing number, $\mathrm{rt}(G)$; see (6)) where $|V| = n$. A (Δ, D)-tree has max degree Δ and diameter D. The generalized hierarchical product of the graphs G_1 and $G_2 = (V_2, E_2)$ is denoted by $\Pi_{\vec{v}}(G_1, G_2)$, for $\vec{v} \in \{0,1\}^{|V_2|}$ (see Definition 2). The special cases of the Cartesian product of graphs, the r-dimensional grid, and the modular graph are also listed. Let $\vec{1} \coloneqq [1 \ldots 1]$ and $\vec{e}_1 \coloneqq [1\,0 \ldots 0]$.

	Worst-case circuit depth	
Graph family	Sorting (comparators)	Routing nr. (transpositions)
path (P_n)	n [25]	n [2]
complete (K_n)	$\Theta(\log n)$ [1]	2 [2]
(Δ, D)-tree	$O(\min(\Delta, \log(n/D))\, n)$ [4]	$3n/2 + O(\log n)$ [56]
$\Pi_{\vec{v}}(G_1, G_2)$	not known	$\left\lceil \frac{\lvert V_2 \rvert}{\mathrm{ham}(\vec{v})} \right\rceil (\mathrm{rt}(G_1) + \mathrm{rt}(G_2)) + \mathrm{rt}(G_2)$
$G_1 \times G_2 = \Pi_{\vec{1}}(G_1, G_2)$	not known	$2\,\mathrm{rt}(G_1) + \mathrm{rt}(G_2)$ [2]
$\bigtimes_{i=1}^{r} P_{n_i}$	$n_1 + 2\sum_{i=2}^{r} n_i + o(\cdot)$ [26]	$n_1 + 2\sum_{i=2}^{r} n_i$ [2]
$\Pi_{\vec{e}_1}(K_{n_1}, K_{n_2})$	not known	$3n_2 + 2$

We also propose using Token Swapping [55] for minimizing the total number of SWAPs, which is relevant when optimizing for total circuit size (Section 3.2). We generalize this problem to partial permutations and obtain a 4-approximation algorithm (Theorem 7).

Finally, we evaluate our circuit transformations on large quantum circuits (Section 5) and compare their performance with the circuit transformation included in the Qiskit software (Section 2.2.1) [8]. We find that the relative performance varies significantly with the circuit type and architecture. When minimizing circuit size, the greedy swap circuit transformation is one of the best, though some improvement may be gained using some of our specialized circuit transformations. For depth, some of our specialized circuit transformations do best on random circuits on grid architectures, whereas Qiskit's circuit transformation does well on modular architectures. For quantum signal processing circuits [32] we find that the depth is best minimized by our greedy swap circuit transformation.

2 Constructing Circuit Transformations

Program transformations are algorithms that modify computer programs while retaining functionality [42]. In a similar vein, we define a *circuit transformation* as an algorithm that modifies an input quantum circuit to produce an output quantum circuit with the same functionality. We represent an architecture by a simple graph $G = (V, E)$, and let Q denote the set of qubits of the input circuit. A circuit transformation is *architecture-respecting* if it produces injective initial and final mappings of the form $\hat{p}\colon Q \to V$ and an architecture-respecting output circuit. The output circuit is architecture-respecting if for each two-qubit gate acting on (qubit) vertices v_1, v_2 we have $(v_1, v_2) \in E$ (where the ordering is irrelevant since G is undirected). Henceforth, we only consider circuit transformations that are architecture-respecting, and we refer to them simply as circuit transformations. We propose a construction for a general circuit transformation that may use the properties of the underlying architecture by relying on a specialized subroutine for moving qubits called a *permuter* (Section 3), and a subroutine determining where to place qubits, called a *mapper* (Section 4). We show in Appendix C that our circuit transformations are polynomial-time in the circuit size and architecture graph size.

To be able to transform a circuit, we must have $|Q| \leq |V|$, and the output circuit must contain a qubit for every vertex in the architecture. Throughout the circuit transformation, we keep track of the injective current placement of qubits $\hat{p}\colon Q \to V$. The initial and final

values of $\hat{p}$ are also the initial and final mappings, respectively, of qubits to the architecture. A gate is *executed* by appending it to the output circuit. Two-qubit gates with qubits $q_1, q_2 \in Q$ can only be executed when $(\hat{p}(q_1), \hat{p}(q_2)) \in E$. By adding SWAP gates to the output circuit, we can change $\hat{p}$ and thereby unitarily transform quantum circuits for execution on an architecture.

2.1 Definitions

Partial Functions and Partial Permutations. For sets X and Y, a *partial function* $f\colon X \rightharpoonup Y$ is a mapping from $\mathrm{dom}(f) \subseteq X$ to $\mathrm{image}(f) \coloneqq \{f(x) \mid x \in \mathrm{dom}(f)\} \subseteq Y$. However, $f(x)$ is undefined for $x \in X \setminus \mathrm{dom}(f)$. We consider such elements x *unmapped*. For $x \in \mathrm{dom}(f)$, we write $x \mapsto f(x)$ and say that x is *mapped to* $f(x)$. We can then define any partial function f as a set of mappings, $f \coloneqq \{x \mapsto y \mid x \in X, y \in Y\}$, where all preimages must be distinct (i.e., if $x \mapsto y \in f$ and $x' \mapsto y' \in f$ with $y \neq y'$, then $x \neq x'$). A *total* function $\hat{f}$ is a partial function where $\mathrm{dom}(\hat{f}) = X$ and is denoted $\hat{f}\colon X \to Y$. By the term "function" we will mean a total function.

A partial function f is *injective* iff $\forall x, x' \in \mathrm{dom}(f)$ with $x \neq x', f(x) \neq f(x')$. A function $\hat{f}\colon X \to Y$ is *surjective* iff $\forall y \in Y, \exists x \in X : f(x) = y$. A *bijective partial function* f is a partial function that is injective and is denoted $f\colon X \rightharpoonup Y$ (note that such an f is necessarily surjective on its image). A *bijective function* $\hat{f}$ is both injective and surjective and is denoted by $\hat{f}\colon X \leftrightarrow Y$. For any bijective (partial) function f there exists an inverse function $f^{-1}\colon \mathrm{image}(f) \to \mathrm{dom}(f)$.

A partial permutation π is any bijective partial function with the same domain and codomain, i.e., $\pi\colon X \rightharpoonup X$. Similarly, a total permutation is any $\sigma\colon X \leftrightarrow X$. By "permutation" we mean a total permutation.

We also define some notions specifically useful for this paper. An *unmapped vertex* is a vertex in $V \setminus \mathrm{dom}(\pi)$, for a graph $G = (V, E)$ and $\pi\colon V \rightharpoonup V$. We define the union of partial functions $f\colon X \rightharpoonup Y$ and $g\colon X \rightharpoonup Y$ when $\mathrm{dom}(f) \cap \mathrm{dom}(g) = \emptyset$ as

$$(f \cup g)(x) \coloneqq \begin{cases} f(x) & \text{if } x \in \mathrm{dom}(f)\,, \\ g(x) & \text{if } x \in \mathrm{dom}(g)\,. \end{cases} \tag{1}$$

Furthermore, $(f \cup g)$ is a bijective partial function iff f and g are bijective partial functions and $\mathrm{image}(f) \cap \mathrm{image}(g) = \emptyset$. A *completion* of $\pi\colon X \rightharpoonup X$ is a $\hat{\pi}\colon X \leftrightarrow X = (\pi \cup \sigma)$ for some $\sigma\colon X \rightharpoonup X$, where $\mathrm{dom}(\sigma) = X \setminus \mathrm{dom}(\pi)$ and $\mathrm{image}(\sigma) = X \setminus \mathrm{image}(\pi)$.

Directed Acyclic Graph Representation of a Circuit. A quantum circuit can be viewed as a directed acyclic graph (DAG), where vertices represent gates and directed edges represent qubit dependencies. We define the first *layer* of the DAG, L, to be the set of all vertices without predecessors. By removing L and taking the first layer of the resulting DAG, we can define the second layer, and so on.

The size of a circuit is the number of gates it contains (i.e., the number of vertices in the DAG); the depth of a circuit is the number of layers. It is natural to minimize either the depth, corresponding to the execution time when gates can be applied in parallel, or the size, corresponding to the total number of operations that must be performed. We are mostly only interested in two-qubit gates and their qubits. Therefore, let us define $\mathrm{tg}\colon V_D \rightharpoonup Q \times Q$, where V_D is the set of DAG vertices, that outputs the pair of qubits acted on by the DAG vertex, for two-qubit gates. For simplicity, we denote $\mathrm{tg}(L) \coloneqq \{\mathrm{tg}(g) \mid g \in L, g \text{ is a two-qubit gate}\}$.

2.2 Architecture-Respecting Circuit Transformations

We now describe some specific architecture-respecting circuit transformations. We first describe two basic circuit transformations, one provided by the Qiskit software (Section 2.2.1) and another that uses a simple greedy approach (Section 2.2.2). Then, in Section 2.2.3 we specify a family of circuit transformations that builds on specialized procedures for qubit placement and routing.

2.2.1 Qiskit Circuit Transformation

The open-source quantum computing software framework Qiskit [8] contains a circuit transformation[1] that we build upon in one of our mappers (Section 4). We specify this transformation here and compare it with our other approaches to circuit transformations in Section 5.

We initialize $\hat{p}$ arbitrarily. Fix a number of trials, $k \in \mathbb{N}$, for each layer. We do the following in trial $i \in [k]$ where $[k] \coloneqq \{1, \ldots, k\}$: For all $v, u \in V$, sample a symmetric weight $d_i(v,u) \coloneqq (1 + \mathcal{N}(0, 1/N))\, d(v,u)^2$ independently for $(v,u) \in V \times V$, where $\mathcal{N}(\mu, \sigma)$ represents a sample from the normal distribution with mean $\mu \in \mathbb{R}$ and standard deviation $\sigma \geq 0$, and $d\colon V \times V \to \mathbb{N}$ is the shortest distance function on the architecture graph. We define an objective function as the sum of gate distances,

$$S \coloneqq \sum_{(q_1,q_2)\in \mathrm{tg}(L)} d_i(\hat{p}(q_1), \hat{p}(q_2))\,. \tag{2}$$

We now try to SWAP pairs of qubits to decrease S. Specifically, we construct a set of SWAPs by iterating over all edges $e \in E$ and greedily adding the corresponding SWAP if it decreases S and neither endpoint of e is already involved in some SWAP. We execute the set of SWAPs and update S. We then iterate this process until either $S = |\mathrm{tg}(L)|$; or there is no SWAP that decreases S; or we reach the upper bound of $2|V|$ iterations.

Now, if $S = |\mathrm{tg}(L)|$ then the algorithm has successfully found a sequence of SWAPs and all gates in L can be executed. The result of trial i is then set to this sequence of SWAPs. Otherwise, trial i is a failure. If there is at least one successful trial out of k trials, we execute the SWAPs of a successful trial with the fewest SWAPs and then execute all gates in L.

If no trial was successful, we apply the same routine for finding SWAPs that minimize S, but taking only a single gate $(q_1, q_2) \in \mathrm{tg}(L)$ at a time. Note that this results in a sequence of SWAPs along the shortest path between $\hat{p}(q_1)$ and $\hat{p}(q_2)$. After each such step we execute the selected gate. We repeat this until all gates in $\mathrm{tg}(L)$ have been executed and also execute all single-qubit gates in L. Finally, we remove the vertices in L from the input circuit DAG and iterate this process until all gates in the input circuit are executed.

2.2.2 Greedy Swap Circuit Transformation

We also describe a simple greedy approach to circuit transformations. Similar to the Qiskit circuit transformation described above, we prioritize SWAPs that maximally reduce the total distance between the qubits $\mathrm{tg}(L)$, but now using the simpler objective function

$$R \coloneqq \sum_{(q_1,q_2)\in \mathrm{tg}(L)} d(\hat{p}(q_1), \hat{p}(q_2))\,. \tag{3}$$

Note that this is different from (2), where a randomized distance d_i is used.

[1] We base our description on `qiskit.mapper.swap_mapper` from Qiskit version 0.6.1.

We construct an initial $\hat{p}$ as follows. Let us consider the first layer L' of the circuit consisting of only two-qubit gates (i.e., single-qubit gates are ignored), initialize $p' \colon Q \rightharpoonup V$ as undefined everywhere, and set $U := \emptyset \subseteq V$. We iteratively construct

$$p' \leftarrow p' + \{q_1 \mapsto v_1, q_2 \mapsto v_2 \mid (q_1, q_2) \in L', (v_1, v_2) \in M\}, \tag{4}$$

where $M \subseteq E$ is a maximum matching of G, remove (q_1, q_2) from L', set $U \leftarrow U \cup \{v_1, v_2\}$, and recompute M on the subgraph of G with the vertices $V \setminus U$.[2] The remaining qubits $Q \setminus \mathrm{dom}(p')$ are arbitrarily mapped to the available vertices $V \setminus \mathrm{image}(p')$ to obtain $\hat{p}$.

In every iteration, we construct a set of disjoint gates to execute. We first execute as many gates from L as possible given $\hat{p}$, and we remove these gates from the input circuit. Second, let E_i, for $i \in [2]$, be the set of edges where executing a SWAP would decrease R by i, excluding edges which already had a vertex involved in a gate this iteration. We then greedily execute gates from E_2 first and E_1 second, updating both E_is as we go. If we were not able to execute a gate from L and no SWAPs were executed, then, as a fallback, we deterministically pick a two-qubit gate $(q_1, q_2) \in \mathrm{tg}(L)$ and SWAP along the first edge on the shortest path between $\hat{p}(q_1)$ and $\hat{p}(q_2)$. We update $\hat{p}$ according to the inserted SWAPs, update L, and finally update R. This process is repeated until the input circuit is empty.

The fallback routine ensures that this circuit transformation always produces an output circuit. The value R strictly decreases in every iteration until a gate can be executed unless the fallback routine is performed, in which case R stays the same. On repeated calls to the fallback routine, the same two-qubit gate is picked deterministically until it is executed. This happens within $\mathrm{diam}(G) + 1$ iterations, where $\mathrm{diam}(G)$ denotes the diameter of G. By induction we see that the whole circuit will be executed.

2.2.3 Constructing Architecture-aware Circuit Transformations

We now present our construction for a general circuit transformation and make some definitions more precise. Let a *permuter* (Section 3) be a subroutine that, given $\pi \colon V \rightharpoonup V$, outputs a sequence of transpositions that implements π while respecting the architecture constraints. Let a *mapper* (Section 4) be a subroutine that, given $\hat{p}$, a permuter, and a quantum circuit, computes a new placement of qubits, $p \colon Q \rightharpoonup V$, such that some gates of the input circuit can be executed.

Initialize $\hat{p}$ in the same way as the greedy swap circuit transformation. We repeat the following steps until the entire circuit has been transformed:

1. Use the given mapper to find a placement, $p \colon Q \rightharpoonup V$, for the remaining input circuit;
2. Let "$\circ$" denote partial function composition, i.e., given $g \colon X \rightharpoonup Y$ and $f \colon Y \rightharpoonup Z$, $(f \circ g)(x) := f(g(x))$, for $x \in \mathrm{dom}(g)$ and $g(x) \in \mathrm{dom}(f)$. We use the permuter to find transpositions implementing $p \circ \hat{p}^{-1} \colon V \rightharpoonup V$ and replace the transpositions with SWAP gates to construct a permutation circuit to execute. We also update $\hat{p}$ to reflect the new placement of qubits after running the permutation circuit.
3. Execute all gates in L that can be executed in accordance with $\hat{p}$, remove these gates from the input circuit, and recompute L.

[2] This is equivalent to running the greedy depth mapper (Section 4) on the input circuit with only two-qubit gates, an arbitrary $\hat{p}$, and free permutations of qubits. In other words, the greedy depth mapper will pick a placement of qubits on the architecture unconstrained by movement of qubits, since this is the initial placement.

3 Partial Permutations via Transpositions

In this section we provide routing algorithms for implementing partial permutations via transpositions constrained to edges of a graph. We call such algorithms *permuters*. The Routing via Matchings and Token Swapping problems capture exactly our optimization goals of implementing a permutation of qubits on a quantum architecture while minimizing the circuit depth and size, respectively.

3.1 Partial Routing Via Matchings

The framework of Routing via Matchings captures how to permute qubits on a graph using a circuit of the smallest possible depth [2]. We first define a generalization of Routing via Matchings that allows for partial permutations and then provide permuters for implementing partial permutations for some architectures of interest.

▶ **Definition 1** (Partial Routing via Matchings)**.** Partial Routing via Matchings *is the following optimization problem. Given a simple graph $G = (V, E)$ and a $\pi\colon V \rightharpoonup V$, the objective is to find the smallest $k \in \mathbb{N}$ such that there exist matchings $M_1, \ldots, M_k \subseteq E$ on G, where each matching induces a permutation as a product of disjoint transpositions*

$$\pi_{M_i} = \prod_{(v,u)\in M_i} (v\ u), \quad \textit{such that} \quad \hat{\pi} = \prod_{i=1}^{k} \pi_{M_i} \tag{5}$$

is a completion of π.

Routing via Matchings is the special case of Partial Routing via Matchings where π is constrained to be a (total) permutation. The *partial routing number* of $\pi\colon V \rightharpoonup V$ on G is $\mathrm{rt}(G, \pi) \coloneqq k$, where k obtains the minimum in Definition 1. The *routing number* [2] is the special case of the partial routing number where π is total. In this paper, we simply refer to the partial routing number as the routing number. The routing number of G is defined as

$$\mathrm{rt}(G) \coloneqq \max_{\sigma\in \mathrm{Sym}(V)} \mathrm{rt}(G, \sigma)\,, \tag{6}$$

where we maximize over all permutations $\sigma\colon V \leftrightarrow V$ (here $\mathrm{Sym}(V)$ denotes the group of such permutations). Note that we only optimize over permutations, since for any $\pi\colon V \rightharpoonup V$,

$$\mathrm{rt}(G, \pi) = \min_{\hat{\pi}} \mathrm{rt}(G, \hat{\pi})\,, \tag{7}$$

where we minimize over all completions $\hat{\pi}$ of π.

An alternate way to interpret (Partial) Routing via Matchings is to assign *tokens* to all $v \in \mathrm{dom}(\pi)$ and destinations $\pi(v)$ for the tokens. A token can only by moved through an exchange of tokens between adjacent vertices. The goal is to move all tokens to their destination in as few matchings (specifying exchange locations) as possible. If a vertex does not hold a token at the time of an exchange with a neighbor, as can be the case in Partial Routing via Matchings, then after the exchange the neighbor will not hold a token.

We give simple constructions for permuters of the complete graph, K_n, and the path graph, P_n, for $n \in \mathbb{N}$. Let V be the vertex set of the respective graph and $\pi\colon V \rightharpoonup V$ given. For K_n, if $|\mathrm{dom}(\pi) \cup \mathrm{image}(\pi)| = 2|\mathrm{dom}(\pi)|$ all mappings are disjoint, so we return $\{(v, \pi(v)) \mid v \in \mathrm{dom}(\pi)\}$ as a single matching that implements π. Otherwise, we construct an arbitrary completion $\hat{\pi}$ of π and run the standard algorithm for Routing via Matchings for complete graphs on $\hat{\pi}$ [2], obtaining $\mathrm{rt}(K_n, \pi) \leq 2$.

For P_n, let $V \cong [n]$, ordered from one end of the path to the other (picking ends arbitrarily). Iterate through $i \in V$ in ascending order, setting

$$\hat{\pi}(i) = \begin{cases} \pi(i) & \text{if } i \in \mathrm{dom}(\pi), \\ \min\left(V \setminus \mathrm{image}(\hat{\pi})\right) & \text{otherwise.} \end{cases} . \tag{8}$$

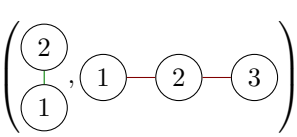
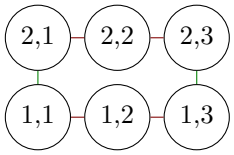
We then run the standard path routing algorithm [2] on $\hat{\pi}$, obtaining $\mathrm{rt}(P_n, \pi) \leq n$. It remains an open question whether a tighter bound can be proven as a function of some property of π.

Hierarchical Product

The *generalized hierarchical product* (henceforth *hierarchical product*) of graphs [6] produces various subgraphs of the Cartesian product of graphs that include natural models of quantum computer architectures [5].

▶ **Definition 2** (Hierarchical Product [6]). *For $j \in \{1, 2\}$, let $G_j = (V_j, E_j)$ be a graph with $n_j \coloneqq |V_j|$ vertices and adjacency matrix $A_j \in \mathcal{M}_{n_j}$, where $\mathcal{M}_k$ is the set of $k \times k$ boolean matrices for $k \in \mathbb{N}$. Then the* hierarchical product $\Pi_{\vec{v}}(G_1, G_2)$*, for $\vec{v} \in \{0,1\}^{n_2}$, has vertex set $V_1 \times V_2$ and adjacency matrix $A_1 \otimes \mathrm{diag}(\vec{v}) + \mathbb{I}_{n_1} \otimes A_2$, where $\mathbb{I}_{n_1} \in M_{n_1}$ is the $n_1 \times n_1$ identity matrix, $M_1 \otimes M_2 \in \mathcal{M}_{n_1 n_2}$ is the Kronecker product of $M_1 \in \mathcal{M}_{n_1}$ and $M_2 \in \mathcal{M}_{n_2}$, and $\mathrm{diag}(\vec{v}) \in M_{n_2}$ is the diagonal matrix with the entries of $\vec{v}$ on the diagonal.*

Intuitively, this graph consists of n_1 copies of G_2, where the jth vertices in all copies of G_2 are connected by a copy of G_1 if $\vec{v}_j = 1$. We restrict ourselves to connected simple graphs, so A_1 and A_2 are symmetric 0–1 matrices and $\vec{v}$ is nonzero. An example of the hierarchical product of two path graphs is

$\Pi_{[1\ 0\ 1]}(P_2, P_3) = \Pi_{[1\ 0\ 1]}$  $=$  (9)

The Cartesian product is $\Pi_{\vec{1}}$, where $\vec{1} \coloneqq [1 \ldots 1]$, and $\Pi_{\vec{e}_i}$ is the rooted product of graphs, rooted at the ith vertex of G_2.

We define the vertex-induced subgraph of any graph $G = (V, E)$ for vertex set $U \subseteq V$ as

$$G[U] \coloneqq (U, E \cap (U \times U)) . \tag{10}$$

Now, let $G = (V, E) = \Pi_{\vec{v}}(G_1, G_2)$ and denote the vertices of G by $v = (v_1, v_2) \in V_1 \times V_2 = V$. We define $\mathcal{G}_i = (\mathcal{V}_i, \mathcal{E}_i) \coloneqq G[\{i\} \times V_2]$, for $i \in V_1$. Note that each $\mathcal{G}_i$ is isomorphic to G_2, so the permuter for G_2 can be used for $\mathcal{G}_i$. We also define the *communicator vertices* of $\mathcal{G}_i$ as the vertices $\{i\} \times \{j \in V_2 \mid \vec{v}_j = 1\} \subseteq \mathcal{V}_i$ and index them in ascending order (for some ordering of V). Note that the jth communicator vertex (of any $\mathcal{G}_i$) also belongs to $G[V_1 \times \{j\}]$, which is isomorphic to G_1.

A useful metric is the maximum number of vertices that need to leave or enter any $\mathcal{G}_i$ to implement π, defined as the degree of π,

$$\deg(\pi) \coloneqq \max \bigcup_{i \in V_1} \left\{ \left|\{v \in \mathrm{dom}(\pi) \cap \mathcal{V}_i \mid \pi(v) \notin \mathcal{V}_i\}\right|, \left|\{v \in \mathrm{dom}(\pi) \setminus \mathcal{V}_i \mid \pi(v) \in \mathcal{V}_i\}\right| \right\} . \tag{11}$$

In every iteration of the routing algorithm, we route a set $R = \{v^{(i)} \in \mathcal{V}_i \mid i \in V_1\}$ such that all $\pi(v)_1$ are distinct, for $v \in R$ and $\pi(v) = (\pi(v)_1, \pi(v)_2) \in V$. Undefined values are always considered distinct. We call such R a set of *representative vertices*, and we view $v^{(i)}$ as the representative vertex of V_i.

Algorithm 3.1: PARTIAL ROUTING VIA MATCHINGS on the hierarchical product of graphs $\Pi_{\vec{v}}(G_1, G_2)$. In Line 1, *routing* means constructing a partial permutation σ on a subgraph (G_1 or G_2), using the applicable permuter to find transpositions implementing σ, and applying those transpositions to update π and each R_i.

input : $\pi\colon V_1 \times V_2 \rightharpoonup V_1 \times V_2$; permuters on G_1 and G_2

1 Let R_i, for $i \in [\deg(\pi)]$, be given by Lemma 3

2 **for** $i = 1, \ldots, \left\lceil \frac{\deg(\pi)}{\mathrm{ham}(\vec{v})} \right\rceil$ **:**

3 **foreach** $j \in V_1$ **:**

4 on $\mathcal{G}_j$, for all $k \in [\mathrm{ham}(\vec{v})]$, route the (unique) vertex $v \in R_{(i-1)\cdot\mathrm{ham}(\vec{v})+k} \cap \mathcal{V}_j$ to the k-th communicator vertex of $\mathcal{G}_j$ `// For` R_ℓ `with` $\ell > \deg(\pi)$`, do nothing`

5 **foreach** communicator vertex (v_1, v_2) of $\mathcal{G}_1$ **:** `// All copies of` G_1

6 on $G[V_1 \times \{v_2\}] = (V', E')$, route each $v \in V' \cap \mathrm{dom}(\pi)$ to $(\pi(v)_1, v_2) \in V'$

7 **foreach** $i \in V_1$ **:**

8 route all $v \in \mathrm{dom}(\pi) \cap V_i$ to $\pi(v)$ within $\mathcal{G}_i$

9 **return** the transpositions that implement this routing

▶ **Lemma 3** (Proof in Appendix A.1). *For a graph $\Pi_{\vec{v}}(G_1, G_2)$, $\pi\colon V \rightharpoonup V$, let $d \coloneqq \deg(\pi)$. We can find distinct sets of representative vertices R_i, for $i \in [d]$, such that*

$$\{v \in \mathrm{dom}(\pi) \mid v_1 \neq \pi(v)_1\} \subseteq \bigcup_{i \in [d]} R_i \,.$$

Algorithm 3.1 specifies a permuter for the hierarchical product. We prove the following performance bounds for this algorithm

▶ **Theorem 4.** *For a graph $\Pi_{\vec{v}}(G_1, G_2)$, Algorithm 3.1 finds a sequence of transpositions that implements $\pi\colon V \rightharpoonup V$ certifying that*

$$\mathrm{rt}(\Pi_{\vec{v}}(G_1, G_2), \pi) \leq \left\lceil \frac{\deg(\pi)}{\mathrm{ham}(\vec{v})} \right\rceil (\mathrm{rt}(G_1) + \mathrm{rt}(G_2)) + \mathrm{rt}(G_2) \,,$$

where $\mathrm{ham}(\vec{v})$ is the Hamming weight of $\vec{v}$, i.e., the number of ones in $\vec{v}$.

Proof. In every round of routing, we route $\mathrm{ham}(\vec{v})$ sets R_i to their destination $\mathcal{G}_j$s, for $j \in V_1$. In each round, we route on all copies of G_2 in parallel and then route on all copies of G_1 in parallel. After routing all R_i in at most $\lceil \deg(\pi)/\mathrm{ham}(\vec{v}) \rceil$ rounds, Lemma 3 ensures that only permutations local to each $\mathcal{G}_j$ remain. Finally, we route vertices to their destinations, as given by π, in each $\mathcal{G}_j$ independently using the permuter for G_2. ◀

As a possible optimization, we can remove some vertices from the partial permutations in the routing steps. For each removed vertex, we must ensure that the remaining steps of the routing algorithm remain valid. Specifically, let there be a $u \in \mathcal{G}_i \cap R_k$ for $i \in V_1$ and $k \in [\deg(\pi)]$. If $u \in \mathrm{dom}(\pi)$ and $\pi(u) \in \mathcal{V}_i$, then we remove it since it does not need to be routed outside of $\mathcal{G}_i$. Otherwise, if $u \notin \mathrm{dom}(\pi)$, we remove it unless $\{R_k \cap \mathrm{dom}(\pi) \mid \pi(v) \in \mathcal{G}_i\} \neq \emptyset$ since an unmapped vertex is expected at the communicator vertex in the second loop of the routing round. We apply this optimization in our implementation of the permuter for modular graphs (Appendix A.2).

We show a lower bound on the routing number of hierarchical products of graphs, which can be shown to be tight up to constant factors (see full arXiv version on title page).

▶ **Theorem 5.** *For a graph* $\Pi_{\vec{v}}(G_1, G_2)$ *and any* $\pi\colon V \rightharpoonup V$,

$$2\left\lceil \frac{\deg(\pi)}{\mathrm{ham}(\vec{v})} \right\rceil - 1 \leq \mathrm{rt}(\Pi_{\vec{v}}(G_1, G_2), \pi)\,.$$

Proof. Let us consider the token-based formulation of Partial Routing via Matchings. At most $\deg(\pi)$ tokens need to be moved out of any $\mathcal{G}_i$, for $i \in V_1$. Every matching can move at most $\mathrm{ham}(\vec{v})$ tokens out of their original $\mathcal{G}_i$. Once moved out, a new set of tokens must be moved onto the $\mathrm{ham}(\vec{v})$ communicator vertices. Therefore, it takes at least $2\lceil \deg(\pi)/\mathrm{ham}(\vec{v})\rceil - 1$ matchings to move $\deg(\pi)$ tokens out of any $\mathcal{G}_i$. ◀

For special cases of interest to quantum architectures, we analyze the modular graph in Appendix A.2 and provide a permuter specialized to Cartesian products of graphs with some heuristic optimizations in the full arXiv version.

3.2 Partial Token Swapping

The Token Swapping problem is similar to Routing via Matchings, but minimizes the total number of transpositions instead of the depth [55]. It follows that the induced permutation circuit is optimized for circuit size. The decision version of Token Swapping was first shown to be NP-complete [38] and, later, hardness was shown in parametrized complexity of the number of swaps k [10]. For $\epsilon > 0$, a $(1+\epsilon)$-approximation algorithm is an algorithm that produces a solution within a factor $(1+\epsilon)$ of optimal for all valid inputs. Here, we define a generalized version of Token Swapping that allows for partial permutations, and then give a 4-approximation algorithm for this problem on connected simple graphs that generalizes a previous 4-approximation algorithm for total permutations [38].

▶ **Definition 6** (Partial Token Swapping)**.** *We define* Partial Token Swapping *as an optimization problem. Given are a graph* $G = (V, E)$ *and partial permutation* $\pi\colon V \rightharpoonup V$*. The objective is to find the smallest* $k \in \mathbb{N}$ *such that* $\hat{\pi} = (u_1\ v_1)(u_2\ v_2)\dots(u_k\ v_k)$*, for* $\hat{\pi}$ *some completion of* π *and* $(u_i, v_i) \in E$ *for* $i \in [k]$*.*

Analogous to the routing number, we define the *routing size* of $\pi\colon V \rightharpoonup V$ on G, $\mathrm{rs}(G, \pi)$, to be the minimum k in Definition 6, and the routing size of G as

$$\mathrm{rs}(G) \coloneqq \max_{\sigma \in \mathrm{Sym}(V)} \mathrm{rs}(G, \sigma)\,. \tag{12}$$

Token Swapping is the special case of Partial Token Swapping where π is constrained to be a total permutation. Partial Token Swapping also has an equivalent token-based formulation, similar to Partial Routing via Matchings.

We now describe a permuter that aims to minimize the circuit size. Miltzow et al. [38] gave a 4-approximation algorithm for Token Swapping. Here, we generalize their results to Partial Token Swapping and prove that our generalized algorithm is also a 4-approximation algorithm. For this section, we consider the token-based formulation of Partial Token Swapping (recall the notion of tokens introduced in Section 3.1).

The main idea of Miltzow et al. is to perform swaps that reduce the sum of all distances of tokens to their destinations. We use the following definitions from [38]: An *unhappy swap* is "an edge swap where one of the tokens swapped is already on its target and the other token reduces its distance to its target vertex (by one)", and a *happy swap chain* is a path of $\ell + 1$ distinct vertices $v_1 v_2 \dots v_\ell$, such that swapping all $(v_i, v_{i+1}) \in E$, for $i \in [\ell - 1]$, in increasing order strictly reduces the distances of all tokens in the chain to their destinations.

Algorithm 3.2: Routing tokens to their destinations while minimizing the number of transpositions. We add an extra step that performs no-token swaps to the algorithm of [38]. For $v \in V$, $N(v) \subseteq V$ denotes the set of neighbors of v. The partial permutation $\mathrm{id}|_{\mathrm{dom}(\pi)} \colon V \rightharpoonup V$ is the restriction of the identity function $\mathrm{id} \colon V \leftrightarrow V$ to $\mathrm{dom}(\pi)$ (so it is undefined outside of $\mathrm{dom}(\pi)$).

input : $\pi \colon V \rightharpoonup V$
while $\pi \neq \mathrm{id}|_{\mathrm{dom}(\pi)}$:
  if there exists a happy swap chain $v_1 v_2 \dots v_\ell$ **then**
    Perform transpositions $(v_1\ v_2)(v_2\ v_3)\dots(v_{\ell-1}\ v_\ell)$
  else if $\exists v \in \mathrm{dom}(\pi), \exists u \in N(v) \setminus \mathrm{dom}(\pi) : d(u, \pi(v)) < d(v, \pi(v))$ **then**
    Perform no-token swap $(v\ u)$ `// u has no token`
  else
    There exists an unhappy swap; perform it
  Update π according to the transpositions that were performed
return The sequence of transpositions that was performed

When considering a partial permutation, not all vertices have a token assigned to them. We add an extra step to the approximation algorithm for Token Swapping to make use of this: Before considering an unhappy swap, we first try to swap a token to a tokenless neighbor if it brings the token closer to its destination. We call this a *no-token swap*. The approximation algorithm for Partial Token Swapping is specified in full in Algorithm 3.2.

▶ **Theorem 7** (Proof in Appendix A.3). *Given a simple connected graph $G = (V, E)$ and $\pi \colon V \rightharpoonup V$, Algorithm 3.2 uses at most $4 \cdot \mathrm{rs}(G, \pi)$ transpositions.*

In the full arXiv version we also show that, when restricted to tree graphs, Algorithm 3.2 is a 3-approximation algorithm or worse, even though it is a 2-approximation algorithm on trees for total permutations [38].

4 Placing Qubits on the Architecture

A *mapping* algorithm (or *mapper*) finds an assignment of circuit qubits to architecture vertices such that gates can be executed efficiently. We specify mappers in terms of the routing number and the routing size. In practice, we replace these quantities with the upper bounds that result from applying our permuters.

Mappers construct *placements* of circuit qubits onto qubits of the architecture. A placement is a bijective partial function $p \colon Q \rightharpoonup V$, where $G = (V, E)$ is the architecture graph. A mapper has access to the *current placement* $\hat{p} \colon Q \to V$ provided by the circuit transformation. Given a placement p and the current placement $\hat{p}$, we can compute a partial permutation $p \circ \hat{p}^{-1} \colon V \rightharpoonup V$ that implements p. All our mappers construct a placement p that is initially undefined everywhere and modify it until finished.

We briefly describe the mappers (and their abbreviations) that we implement and evaluate (see Appendix B for details). We propose mappers optimizing for circuit depth (*depth mappers*) and for circuit size (*size mappers*). For size mappers, specifically, if there is any gate that can be performed without moving qubits, then there is no disadvantage to doing that immediately since it will have to be performed eventually. If there is any such gate, we simply return the empty placement. Thus we assume, for all size mappers, that there are no gates to be performed in-place. Let L be the first layer of gates of the input circuit.

The *greedy depth mapper* (greedy depth) repeatedly places the highest-cost gate in L at its lowest-cost location, where the cost is the routing number to achieve the placement.

The *incremental depth mapper* (incremental) guarantees placement of only the lowest-cost gate, instead of trying to place (almost) all gates in L, and incrementally improves the situation for the other gates.

The *greedy size mapper* (greedy size) is the same as the greedy depth mapper, except that we replace $\mathrm{rt}(\cdot)$ with $\mathrm{rs}(\cdot)$ in the objective function.

The *simple size mapper* (simple) places only the lowest-cost gate at its lowest-cost location.

The *extension size mapper* (extend) first places one gate using the same approach as the simple size mapper. We then try to only place another gate if it is cheaper to place now rather than in a later call to the mapper.

The *Qiskit-based size mapper* (qiskit) is based on Qiskit's circuit transformation. We slightly modify the circuit transformation routine in that it picks edges to SWAP one at a time instead of finding a maximal matching. Since this is a mapper, we only execute one iteration of the circuit transformation: for the first layer L. We return the final placement $\hat{p}$ that would be induced by executing all SWAPs found during the mapping process.

5 Results

We implement the circuit transformation introduced in Section 2.2.3 with a variety of mappers and appropriate permuters. We also implement the greedy swap transformation described in Section 2.2.2. We check the validity of our implementations by testing closeness in fidelity of the original output state and that of the transformed circuit for random input states of 11 qubits on random circuits [47] (described in the next section).

Evaluation Criteria. When testing the performance of these circuit transformations, each is allocated at most 8GB of RAM and 2 days to transform all circuits of a data point. For each data point we transform 10 random circuits and 1 quantum signal processing (QSP) circuit. We consider a 2-day runtime acceptable, given that classical computational resources are plentiful compared to quantum ones. We generate the data on a heterogeneous cluster with Intel Opteron 2354 and Intel Xeon X5560 processors.

The Cartesian permuter (see full arXiv version), the general size permuter (Section 3.2), and Qiskit's circuit transformation (Section 2.2.1) are randomized. We run multiple trials of these permuters and take the best result. Most of the time, trials produce equally good permutation circuits, although occasionally they deviate by a few SWAP gates. Our mappers run permuters $O(|L||E|)$ times, so we do only 4 trials to quickly remove any bad outliers. In contrast, our circuit transformation only directly runs a permuter once per layer of gates, so in this case we perform a slower 100 trials in an attempt to save a few SWAPs. We leave the number of trials for Qiskit's circuit transformation at its default of 40.

We test the performance of circuit transformations for the grid, $P_{n_1} \times P_{n_2}$, using the permuter for Cartesian graphs and for the modular architecture (Appendix A.2), $M_G(n_1, n_2)$, for $n_1, n_2 \in \mathbb{N}$. For an N-qubit circuit, we set $n_1 = n_2 = \lceil \sqrt{N} \rceil$ so that there are enough qubits in the architecture to contain the circuit. By Theorem 4, we know that taking $n_1 = n_2$ minimizes the routing time for our routing strategy among all grids with the same number of qubits. It is less clear how to balance parameters for the modular architecture since Corollary 8 does not depend on n_1 and n_2. For $n_1 \ll n_2$ or $n_2 \ll n_1$, less movement of qubits is needed, since many qubits are adjacent to one another. Thus, we take $n_1 = n_2$ in an attempt to consider a hard case. For some values of N, it may also be possible to find

parameters $n_1' \neq n_2'$ such that $N \leq n_1' n_2' < \lceil \sqrt{N} \rceil^2 = n_1 n_2$, requiring fewer qubits. However, this introduces unwanted size-dependent behavior in our results when $|n_1' - n_2'| \gg 0$ for one circuit size and $n_1' \approx n_2'$ for the next, so we find it preferable to fix $n_1 = n_2$.

We compare the transformed circuits in terms of their *weighted depth* and *weighted size*. For both trapped-ion and superconducting qubits, two-qubit gates typically have longer execution times and lower fidelities than single-qubit gates [31]. Even among two-qubit gates there is a difference between execution times. Assuming fast local unitaries, the SWAP gate has 1–3 times the interaction cost of a CNOT depending on the physical interactions used to realize the gates [51]. For simplicity, we assign unit cost for one-qubit gates, cost 10 for CNOT, and cost 30 for SWAP. We define the weighted size of a circuit as the sum of all gate weights and the weighted depth of a circuit as the maximum-weight path in the DAG of the circuit, where the weight of a path is the sum of the weights of the gates along it.

We consider two circuit families: random circuits and QSP circuits [32]. Random circuits have been proposed for quantum computational supremacy experiments on near-term quantum devices [9, 11]. Such proposals typically construct random circuits so that architecture constraints are automatically obeyed. For our purposes, random circuits provide a class of examples with little structure for circuit transformations to exploit, so we expect them to represent a hard case with large overhead. We generate a fixed set of 10 random circuits of depth 20 for various qubit counts. For each layer, we bin the qubits into pairs uniformly at random and assign each pair of qubits a Haar-random unitary from SU(4). Finally, we decompose each unitary into the smallest possible number of CNOT + SU(2) gates [49]. This random circuit generator is provided by Qiskit [8].

We consider QSP circuits for Hamiltonian simulation as an example of a realistic quantum algorithm. We use the unoptimized circuits provided in [14], decomposed into Z rotations, CNOT gates, and single-qubit Clifford gates. The QSP algorithm requires precise angles that turn out to be expensive to compute. Therefore, [14] uses randomized angles instead, giving a circuit that does not correctly implement the Hamiltonian simulation. Nevertheless, the circuit corresponds to an accurate implementation of QSP, up to rotation angles, and can be used for benchmarking resources. Furthermore, the circuit transformations we construct are unaffected by those angles. We only consider one pair of *phased iterates* of the QSP algorithm ($V_{\phi_i+\pi}^\dagger V_{\phi_{i-1}}$ as in [14, Eq. 31]). A full QSP circuit for the architecture can be constructed by iterating the mapped circuit of such phased iterates, a permutation circuit between iterations, state preparation, and state unpreparation. The cost of the transformed phased iterates dominates all other costs of the construction, so the total cost can be estimated by taking our result times the number of iterations.

The circuit transformations from Section 2.2.3 are constructed from a permuter and a mapper. We denote such circuit transformations by tf: $\{\text{d,s}\} \times M_p$, where M_p is the set of all mappers (referred to by their abbreviations, see Section 4), "d" denotes an appropriate depth permuter (Section 3.1), and "s" denotes the general size permuter (Section 3.2). For example, by tf(d,greedy depth) we denote a circuit transformation with a depth permuter for the architecture and the greedy depth mapper (Section 4).

Numerical Results. Figure 1 plots our results. We first consider the random circuit results. For the grid, we find that tf(d,incremental) shows much slower growth of weighted depth than circuit transformations that do not use depth-optimized permuters (Section 3.1). We also note that tf(d,qiskit) performs much better than Qiskit's circuit transformation (Section 2.2.1), suggesting that depth-optimized permuters can offer a significant advantage. On the modular graph, Qiskit's circuit transformation is much better at minimizing the weighted depth, but

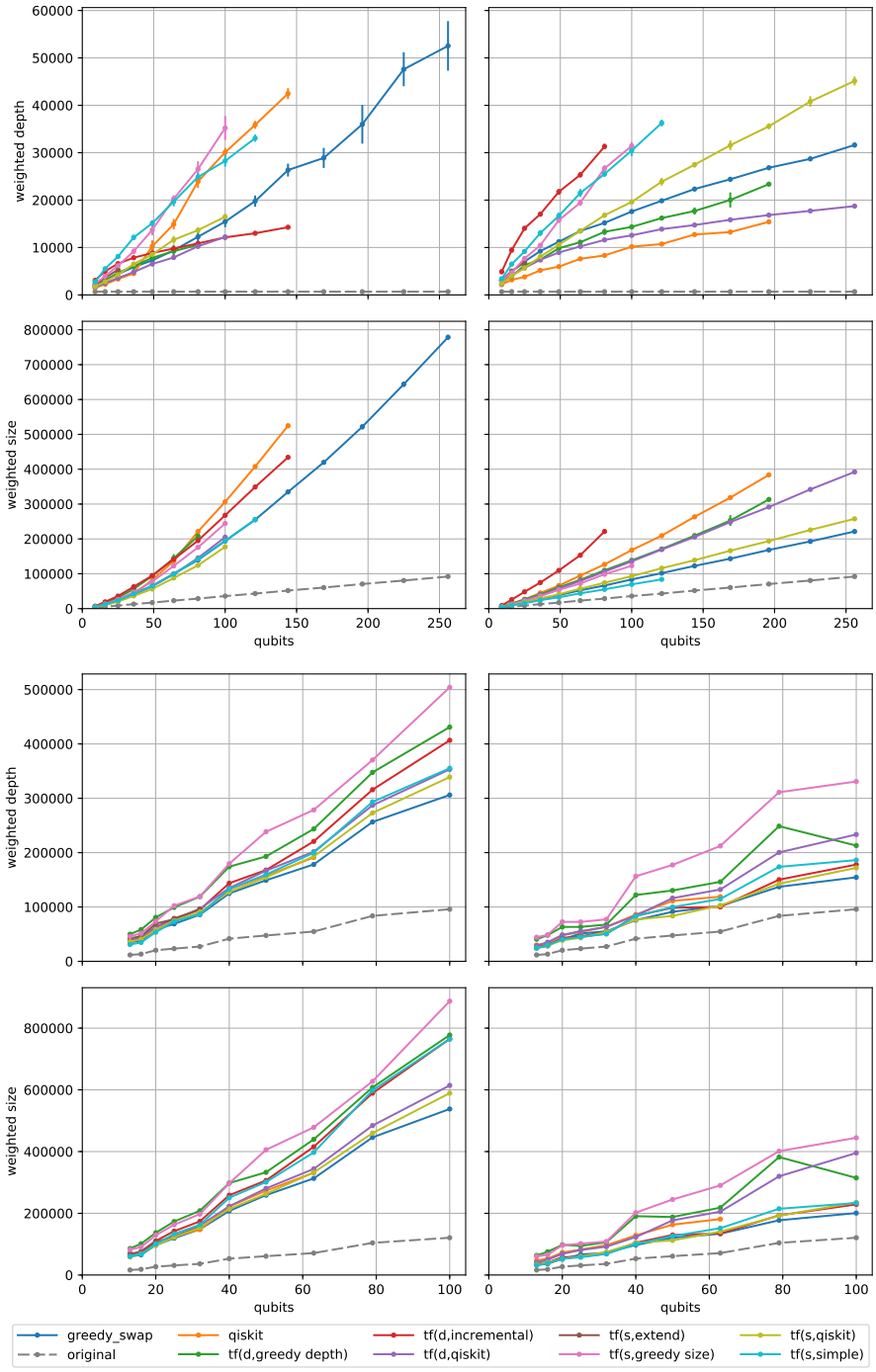


Figure 1 The weighted depth and weighted size for transformed random circuits (top two rows) and QSP circuits (bottom two rows) on the grid architecture (left column) and the modular architecture (right column). We generate fixed sets of 10 random circuits for increasing qubit counts and plot the mean and standard deviation for each data point. One QSP circuit is considered for each data point. The metrics for the original circuit are also given to make the overhead introduced in circuit transformations explicit; note that the original circuit does not respect the architecture constraints. The notation tf: {d,s} $\times$ M_p indicates a circuit transformation constructed from either an appropriate depth ("d") permuter or the size ("s") permuter and one of our mappers (Section 4).

tf(d,qiskit) starts closing the gap for larger sizes. Unfortunately, we do not know if tf(d,qiskit) performs better at larger sizes because Qiskit's circuit transformation is not fast enough to generate the relevant data. Up to 100 qubits tf(s,qiskit) achieves the best weighted size on grid architectures, and tf(s,simple) does best on modular architectures up to 121 qubits. For all sizes the greedy swap circuit transformation (Section 2.2.2) performs as one of the best at optimizing for weighted circuit size. The greedy swap circuit transformation is also able transform larger circuits within the time limit as expected from its lower time complexity.

For larger QSP circuits, the greedy circuit transformation (Section 2.2.2) is the clear winner in both weighted depth and weighted size, suggesting that it may be a good approach for practical quantum circuits. Surprisingly, tf(s,qiskit) also performs fairly well at minimizing the depth despite targeting the circuit size.

6 Future Work

We would like to better understand what circuit transformations work best for which architectures, quantum algorithms, and objective functions. We also would like to use the tools of Partial Routing via Matchings and Partial Token Swapping to establish bounds on the overhead of specific architectures. Ideally, we could use these tools and circuit transformations to design architectures that offer good performance subject to realistic hardware constraints and to compute realistic resource estimates for implementations of quantum algorithms.

There are many ways our methods could be improved. It would be interesting to know whether one can do better than just using SWAP gates to route qubits. Our mapper algorithms may also be improved by including some form of lookahead to consider later layers of the given circuit, or by specializing mappers to particular architectures.

Modeling the architecture as a simple graph loses information about the underlying hardware. For example, in the IBM system the architecture edges have directionality indicating the control and target of CNOTs. In implementations of the modular architecture, the interconnecting links are probably much noisier and slower than local operations. In general, gate costs and times can vary significantly across a hardware implementation and sometimes even vary over time [41]. Adapting to variable costs and keeping track of operations performed asynchronously is challenging but could be worthwhile for architectures that support a mixture of fast and slow operations.

Finally, we hope that future progress on the challenges addressed in this paper will be facilitated by a suitable set of benchmarks of large quantum circuits. We publicly make available and license our source code, benchmark circuits, and results (in `TSV` format) [47] and encourage others to do the same.

References

1 M. Ajtai, J. Komlós, and E. Szemerédi. An $O(n \log n)$ sorting network. In *Proceedings of the fifteenth annual ACM symposium on Theory of computing - STOC '83*. ACM Press, 1983. doi:10.1145/800061.808726.

2 Noga Alon, F. R. K. Chung, and R. L. Graham. Routing Permutations on Graphs via Matchings. *SIAM Journal on Discrete Mathematics*, 7(3):513–530, May 1994. doi:10.1137/s0895480192236628.

3 Indranil Banerjee and Dana Richards. New Results on Routing via Matchings on Graphs. In *Fundamentals of Computation Theory*, pages 69–81. Springer Berlin Heidelberg, 2017. doi:10.1007/978-3-662-55751-8_7.

4 Indranil Banerjee, Dana Richards, and Igor Shinkar. Sorting Networks on Restricted Topologies. In *SOFSEM 2019: Theory and Practice of Computer Science*, pages 54–66. Springer International Publishing, 2019. doi:10.1007/978-3-030-10801-4_6.

5 Aniruddha Bapat, Zachary Eldredge, James R. Garrison, Abhinav Deshpande, Frederic T. Chong, and Alexey V. Gorshkov. Unitary entanglement construction in hierarchical networks. *Physical Review A*, 98(6), 2018. doi:10.1103/PhysRevA.98.062328.

6 L. Barrière, C. Dalfó, M. A. Fiol, and M. Mitjana. The generalized hierarchical product of graphs. *Discrete Mathematics*, 309(12):3871–3881, June 2009. doi:10.1016/j.disc.2008.10.028.

7 R. Beals, S. Brierley, O. Gray, A. W. Harrow, S. Kutin, N. Linden, D. Shepherd, and M. Stather. Efficient distributed quantum computing. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 469(2153), 2013. doi:10.1098/rspa.2012.0686.

8 Luciano Bello, Jim Challenger, Andrew Cross, Ismael Faro, Jay Gambetta, Juan Gomez, Ali Javadi-Abhari, Paco Martin, Diego Moreda, Jesus Perez, Erick Winston, and Chris Wood. Qiskit, 2017. URL: https://www.qiskit.org/.

9 Sergio Boixo, Sergei V. Isakov, Vadim N. Smelyanskiy, Ryan Babbush, Nan Ding, Zhang Jiang, Michael J. Bremner, John M. Martinis, and Hartmut Neven. Characterizing quantum supremacy in near-term devices. *Nature Physics*, 14(6):595–600, 2018. doi:10.1038/s41567-018-0124-x.

10 Édouard Bonnet, Tillmann Miltzow, and Paweł Rzążewski. Complexity of Token Swapping and its Variants. *Algorithmica*, 80(9):2656–2682, October 2017. doi:10.1007/s00453-017-0387-0.

11 Adam Bouland, Bill Fefferman, Chinmay Nirkhe, and Umesh Vazirani. On the complexity and verification of quantum random circuit sampling. *Nature Physics*, October 2018. doi:10.1038/s41567-018-0318-2.

12 Teresa Brecht, Wolfgang Pfaff, Chen Wang, Yiwen Chu, Luigi Frunzio, Michel H. Devoret, and Robert J. Schoelkopf. Multilayer microwave integrated quantum circuits for scalable quantum computing. *npj Quantum Information*, 2(16002), 2016. doi:10.1038/npjqi.2016.2.

13 Stephen Brierley. Efficient Implementation of Quantum Circuits with Limited Qubit Interactions. *Quantum Info. Comput.*, 17(13-14):1096–1104, November 2017.

14 Andrew M. Childs, Dmitri Maslov, Yunseong Nam, Neil J. Ross, and Yuan Su. Toward the first quantum simulation with quantum speedup. *Proceedings of the National Academy of Sciences*, 115(38):9456–9461, 2018. doi:10.1073/pnas.1801723115.

15 Byung-Soo Choi and Rodney van Meter. On the Effect of Quantum Interaction Distance on Quantum Addition Circuits. *ACM Journal on Emerging Technologies in Computing Systems*, 7(3):1–17, August 2011. doi:10.1145/2000502.2000504.

16 Byung-Soo Choi and Rodney van Meter. A $\Theta(\sqrt{N})$-depth Quantum Adder on the 2D NTC Quantum Computer Architecture. *J. Emerg. Technol. Comput. Syst.*, 8(3):24:1–24:22, August 2012. doi:10.1145/2287696.2287707.

17 Robert W. Floyd. Algorithm 97: Shortest path. *Communications of the ACM*, 5(6):345, June 1962. doi:10.1145/367766.368168.

18 Austin G. Fowler, Simon J. Devitt, and Lloyd C. L. Hollenberg. Implementation of Shor's Algorithm on a Linear Nearest Neighbour Qubit Array. *Quant. Info. Comput. 4, 237-251 (2004)*, 2004.

19 Google Quantum AI Lab. A Preview of Bristlecone, Google's New Quantum Processor. URL: https://ai.googleblog.com/2018/03/a-preview-of-bristlecone-googles-new.html.

20 Jeff Heckey, Shruti Patil, Ali JavadiAbhari, Adam Holmes, Daniel Kudrow, Kenneth R. Brown, Diana Franklin, Frederic T. Chong, and Margaret Martonosi. Compiler Management of Communication and Parallelism for Quantum Computation. In *Proceedings of the Twentieth International Conference on Architectural Support for Programming Languages and Operating Systems - ASPLOS '15*. ACM Press, 2015. doi:10.1145/2694344.2694357.

21 Steven Herbert. On the depth overhead incurred when running quantum algorithms on near-term quantum computers with limited qubit connectivity.

22 Yuichi Hirata, Masaki Nakanishi, Shigeru Yamashita, and Yasuhiko Nakashima. An Efficient Method to Convert Arbitrary Quantum Circuits to Ones on a Linear Nearest Neighbor Architecture. In *2009 Third International Conference on Quantum, Nano and Micro Technologies*. IEEE, February 2009. doi:10.1109/icqnm.2009.25.

23 John E. Hopcroft and Richard M. Karp. An $n^{5/2}$ Algorithm for Maximum Matchings in Bipartite Graphs. *SIAM Journal on Computing*, 2(4):225–231, December 1973. doi:10.1137/0202019.

24 IBM Q Team. IBM Q Experience Devices, 2018. URL: https://quantumexperience.ng.bluemix.net/qx/devices.

25 Donald E. Knuth. *Networks for Sorting*, volume 3 of *The Art of Computer Programming*, pages 219–247. Addison-Wesley Professional, second edition, 1998.

26 Manfred Kunde. Optimal sorting on multi-dimensionally mesh-connected computers. In *STACS 87*, pages 408–419. Springer-Verlag, 1987. doi:10.1007/bfb0039623.

27 Samuel A. Kutin, David Petrie Moulton, and Lawren M. Smithline. Computation at a distance. *Chicago Journal of Theoretical Computer Science*, 13(1):1–17, 2007. doi:10.4086/cjtcs.2007.001.

28 L. Lao, B. van Wee, I. Ashraf, J. van Someren, N. Khammassi, K. Bertels, and C. G. Almudever. Mapping of lattice surgery-based quantum circuits on surface code architectures. *Quantum Science and Technology*, 4(1):015005, September 2018. doi:10.1088/2058-9565/aadd1a.

29 Gushu Li, Yufei Ding, and Yuan Xie. Tackling the Qubit Mapping Problem for NISQ-Era Quantum Devices.

30 Chia-Chun Lin, Susmita Sur-Kolay, and Niraj K. Jha. PAQCS: Physical design-aware fault-tolerant quantum circuit synthesis. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 23(7):1221–1234, July 2015. doi:10.1109/tvlsi.2014.2337302.

31 Norbert M. Linke, Dmitri Maslov, Martin Roetteler, Shantanu Debnath, Caroline Figgatt, Kevin A. Landsman, Kenneth Wright, and Christopher Monroe. Experimental comparison of two quantum computing architectures. *Proceedings of the National Academy of Sciences*, 114(13):3305–3310, March 2017. doi:10.1073/pnas.1618020114.

32 Guang Hao Low and Isaac L. Chuang. Optimal Hamiltonian Simulation by Quantum Signal Processing. *Physical Review Letters*, 118(1), January 2017. doi:10.1103/physrevlett.118.010501.

33 Aaron Lye, Robert Wille, and Rolf Drechsler. Determining the minimal number of swap gates for multi-dimensional nearest neighbor quantum circuits. In *The 20th Asia and South Pacific Design Automation Conference*. IEEE, January 2015. doi:10.1109/aspdac.2015.7059001.

34 D. Maslov, S. M. Falconer, and M. Mosca. Quantum Circuit Placement. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 27(4):752–763, 2008. doi:10.1109/tcad.2008.917562.

35 Dmitri Maslov. Linear depth stabilizer and quantum Fourier transformation circuits with no auxiliary qubits in finite-neighbor quantum architectures. *Physical Review A*, 76(5), November 2007. doi:10.1103/physreva.76.052310.

36 Tzvetan S. Metodi, Darshan D. Thaker, Andrew W. Cross, Frederic T. Chong, and Isaac L. Chuang. Scheduling physical operations in a quantum information processor. In Eric J. Donkor, Andrew R. Pirich, and Howard E. Brandt, editors, *Quantum Information and Computation IV*, page 6244. SPIE, 2006. doi:10.1117/12.666419.

37 Silvio Micali and Vijay V. Vazirani. An $O(\sqrt{|V|}|E|)$ algoithm for finding maximum matching in general graphs. In *21st Annual Symposium on Foundations of Computer Science (sfcs 1980)*. IEEE, October 1980. doi:10.1109/sfcs.1980.12.

38 Tillmann Miltzow, Lothar Narins, Yoshio Okamoto, Günter Rote, Antonis Thomas, and Takeaki Uno. Approximation and Hardness for Token Swapping. In Piotr Sankowski and Christos Zaroliagis, editors, *Annual European Symposium on Algorithms*, volume 24, pages 185:1–185:15. Leibniz International Proceedings in Informatics (LIPIcs), 2016.

39 C. Monroe and J. Kim. Scaling the Ion Trap Quantum Processor. *Science*, 339(6124):1164–1169, 2013. doi:10.1126/science.1231298.

40 C. Monroe, R. Raussendorf, A. Ruthven, K. R. Brown, P. Maunz, L.-M. Duan, and J. Kim. Large-scale modular quantum-computer architecture with atomic memory and photonic interconnects. *Physical Review A*, 89(2), February 2014. doi:10.1103/physreva.89.022317.

41 Prakash Murali, Jonathan M. Baker, Ali Javadi Abhari, Frederic T. Chong, and Margaret Martonosi. Noise-Adaptive Compiler Mappings for Noisy Intermediate-Scale Quantum Computers, 2019.

42 Flemming Nielson, Hanne Riis Nielson, and Chris Hankin. *Principles of Program Analysis*. Springer Berlin Heidelberg, 1999. doi:10.1007/978-3-662-03811-6.

43 M. Pedram and A. Shafaei. Layout Optimization for Quantum Circuits with Linear Nearest Neighbor Architectures. *IEEE Circuits and Systems Magazine*, 16(2):62–74, 2016. doi:10.1109/MCAS.2016.2549950.

44 Rigetti. QPU Specifications, 2018. URL: https://www.rigetti.com/qpu.

45 David J. Rosenbaum. Optimal Quantum Circuits for Nearest-Neighbor Architectures. In Simone Severini and Fernando Brandao, editors, *8th Conference on the Theory of Quantum Computation, Communication and Cryptography (TQC 2013)*, volume 22 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 294–307, Dagstuhl, Germany, 2013. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. doi:10.4230/LIPIcs.TQC.2013.294.

46 Mehdi Saeedi, Robert Wille, and Rolf Drechsler. Synthesis of quantum circuits for linear nearest neighbor architectures. *Quantum Information Processing*, 10(3):355–377, June 2011. doi:10.1007/s11128-010-0201-2.

47 Eddie Schoute, Cem Unsal, and Andrew Childs. arct, 2019. URL: https://gitlab.umiacs.umd.edu/amchilds/arct.

48 Alireza Shafaei, Mehdi Saeedi, and Massoud Pedram. Qubit placement to minimize communication overhead in 2D quantum architectures. In *2014 19th Asia and South Pacific Design Automation Conference (ASP-DAC)*. IEEE, January 2014. doi:10.1109/aspdac.2014.6742940.

49 Farrokh Vatan and Colin Williams. Optimal quantum circuits for general two-qubit gates. *Physical Review A*, 69(3), 2004. doi:10.1103/physreva.69.032315.

50 Davide Venturelli, Minh Do, Eleanor Rieffel, and Jeremy Frank. Compiling quantum circuits to realistic hardware architectures using temporal planners. *Quantum Science and Technology*, 3(2):025004, February 2018. doi:10.1088/2058-9565/aaa331.

51 G. Vidal, K. Hammerer, and J. I. Cirac. Interaction Cost of Nonlocal Gates. *Physical Review Letters*, 88(23), 2002. doi:10.1103/physrevlett.88.237902.

52 Mark Whitney, Nemanja Isailovic, Yatish Patel, and John Kubiatowicz. Automated generation of layout and control for quantum circuits. In *Proceedings of the 4th international conference on Computing frontiers - CF '07*, pages 83–94. ACM Press, 2007. doi:10.1145/1242531.1242546.

53 Robert Wille, Oliver Keszocze, Marcel Walter, Patrick Rohrs, Anupam Chattopadhyay, and Rolf Drechsler. Look-ahead schemes for nearest neighbor optimization of 1D and 2D quantum circuits. In *2016 21st Asia and South Pacific Design Automation Conference (ASP-DAC)*. IEEE, January 2016. doi:10.1109/aspdac.2016.7428026.

54 Robert Wille, Aaron Lye, and Rolf Drechsler. Exact Reordering of Circuit Lines for Nearest Neighbor Quantum Architectures. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 33(12):1818–1831, December 2014. doi:10.1109/tcad.2014.2356463.

55 Katsuhisa Yamanaka, Erik D. Demaine, Takehiro Ito, Jun Kawahara, Masashi Kiyomi, Yoshio Okamoto, Toshiki Saitoh, Akira Suzuki, Kei Uchizawa, and Takeaki Uno. Swapping Labeled Tokens on Graphs. In *Lecture Notes in Computer Science*, pages 364–375. Springer International Publishing, 2014. doi:10.1007/978-3-319-07890-8_31.

56 Louxin Zhang. Optimal Bounds for Matching Routing on Trees. *SIAM Journal on Discrete Mathematics*, 12(1):64–77, January 1999. doi:10.1137/s0895480197323159.

57 Alwin Zulehner, Alexandru Paler, and Robert Wille. An Efficient Methodology for Mapping Quantum Circuits to the IBM QX Architectures. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2018. doi:10.1109/tcad.2018.2846658.

58 Alwin Zulehner and Robert Wille. Compiling SU(4) quantum circuits to IBM QX architectures. In *Proceedings of the 24th Asia and South Pacific Design Automation Conference on - ASPDAC '19*, pages 185–190. ACM Press, 2019. doi:10.1145/3287624.3287704.

A Partial Permutations via Transpositions

Here we present proofs and statements that were omitted in the main text of Section 3.

A.1 Hierarchical Product

Proof of Lemma 3. Let $G = (U, V, E)$ be a bipartite multi-graph, with $U = V \coloneqq [n_1]$ the left and right vertex sets, and the edge multi-set

$$E = \{(v_1, \pi(v)_1) \mid v \in \mathrm{dom}(\pi)\}\,. \tag{13}$$

Each vertex $k \in U$ belongs to at most d edges (k, l), for $l \in V$ and $k \neq l$, and each vertex $l' \in V$ belongs to at most d edges (k', l'), for $k' \in U$ and $k' \neq l'$. However, for any $k \in U$ there could be as many as n_2 edges (k, k). For all $k \in U$ we remove as many $(k, k) \in E$ as necessary to ensure that the maximum degree of any vertex in G is d.

We make G d-regular by repeating the following: If $\nexists k \in U$ with $\deg(k) < d$ we are done. Otherwise, such a k exists and $\exists k' \in V$ with $\deg(k') < d$, since

$$\sum_{k \in U} \deg(k) = \sum_{k' \in V} \deg(k')\,. \tag{14}$$

It follows that there exist vertices $u \in \mathcal{V}_k \setminus \mathrm{dom}(\pi)$ and $v \in \mathcal{V}_{k'} \setminus \mathrm{image}(\pi)$. For the purposes of this proof, we set $\pi(u) = v$, effectively adding an edge (k, k') to E.

Now we have modified π so that G is d-regular. By Hall's marriage theorem, there exists a perfect matching in G, and removing it results in a $(d-1)$-regular graph. We iterate this to find d distinct perfect matchings in G. Each edge $(k, k') \in E$ corresponds to some $v \in \mathcal{V}_k$ and $u \in \mathcal{V}_{k'}$, with $\pi(v) = u$. Therefore, each perfect matching corresponds to a set of representative vertices, R_i. Since all perfect matchings are distinct, and all $e \in E$ are covered by some matching, the Lemma follows. ◀

A.2 Modular Graph

Large-scale quantum computation may benefit from a modular design, with many interconnected subunits [39, 40, 12]. As a simple model of a modular quantum processor consisting of n_1 modules with n_2 qubits each, we consider the *modular graph* $M_G(n_1, n_2) \coloneqq \Pi_{\vec{e}_1}(K_{n_1}, K_{n_2}) = (V, E)$, where $\vec{e}_i \in \{0,1\}^{n_2}$, for $i \in [n_2]$, is the ith standard basis vector. In this architecture, two qubits in the same module can be directly coupled, and any two modules can be coupled through their unique communicator qubits. With one minor modification to Theorem 4, we get the following bounds on the routing number of the modular graph.

▶ **Corollary 8.** *For $n_1, n_2 \in \mathbb{N}$ and $\pi \colon V \rightharpoonup V$, we have* $\deg(\pi) \leq \mathrm{rt}(M_G(n_1, n_2), \pi) \leq 3\deg(\pi) + 2$.

Proof. Directly applying Theorem 4 gives $\mathrm{rt}(M_G(n_1, n_2), \pi) \leq 4\deg(\pi) + 2$. However, only one token needs to be routed to the communicator vertex in every round of Algorithm 3.1. We note that this can be routed with one set of parallel transpositions (Section 3.1), saving us one matching every round. To show the lower bound, we apply Theorem 5 with $\mathrm{ham}(\vec{e}_1) = 1$. ◀

A.3 Partial Token Swapping

Proof of Theorem 7. The proof is very similar to [38, Theorem 7] with some minor modifications to account for no-token swaps. Let

$$S \coloneqq \sum_{v \in \mathrm{dom}(\pi)} d(v, \pi(v)) \,, \tag{15}$$

and let s_u, s_h, s_{nt} be the number of unhappy, happy, and no-token swaps, respectively. We know that $\mathrm{rs}(G,\pi) \geq S/2$ since each swap can only reduce S by two. A no-token swap reduces S by one. A happy swap chain of length ℓ reduces S by $\ell + 1$. As such, over the course of the algorithm, $s_h + s_{nt} \leq S$. For an unhappy swap, the token that is swapped away from its destination must next be involved in a happy swap or a no-token swap, so $s_u \leq s_h + s_{nt}$. Thus, we get $s_u + s_h + s_{nt} \leq 4 \cdot \mathrm{rs}(G,\pi)$ as desired. ◀

B Specifics of Mappers

Here, we describe our mappers (Section 4) in full. Let M be a maximum matching in the architecture graph.

B.1 Greedy Depth Mapper

The *greedy depth mapper* iteratively places the highest-cost gate at its lowest-cost location, where cost is measured in terms of the routing number to achieve the placement. More precisely, we initialize the set of used vertices $U \coloneqq \emptyset$ and find a placement $p' \coloneqq \{q_1 \mapsto v_1, q_2 \mapsto v_2\}$ that attains the optimum

$$\max_{(q_1,q_2)\in \mathrm{tg}(L)} \min_{(v_1,v_2)\in M} \mathrm{rt}\big(G, (p \cup \{q_1 \mapsto v_1, q_2 \mapsto v_2\}) \circ \hat{p}^{-1}\big) \,, \tag{16}$$

where we consider both orderings of edges from M, $(v,u),(u,v) \in M$, since edges are undirected. Then, we update $U \leftarrow U \cup \mathrm{dom}(p')$ and recompute M for the graph $G[V \setminus U]$ (recall (10)); we remove the gate associated to (q_1, q_2) from L; we set $p \leftarrow p \cup p'$; and we iterate until $\mathrm{tg}(L) = \emptyset$ or $M = \emptyset$. Finally, we return the placement p.

B.2 Incremental Depth Mapper

Instead of trying to place (almost) all gates in L, the *incremental depth mapper* guarantees placement of only the lowest-cost gate, as given by the routing number, and incrementally improves the situation for the other gates. Specifically, we first find a placement $p_{\min} \coloneqq \{q_1 \mapsto v_1, q_2 \mapsto v_2\}$ that attains the optimum

$$c'_{\min} \coloneqq \min_{(q_1,q_2)\in \mathrm{tg}(L)} \min_{(v_1,v_2)\in E} \mathrm{rt}(G, (p \cup \{q_1 \mapsto v_1, q_2 \mapsto v_2\}) \circ \hat{p}^{-1}) \,, \tag{17}$$

where we consider both orderings of E, $(u,v),(v,u) \in E$. We set $p \leftarrow p_{\min}$ and define $U \coloneqq \{u, v\}$. Let $c_{\min} \coloneqq \max\{c'_{\min}, 1\}$.

We find a placement for the remaining two-qubit gates that (individually) does not exceed $c_{\min}$. We iterate in arbitrary order over $(q_1, q_2) \in \mathrm{tg}(L)$ and do the following: For $i \in [2]$, we construct a set of eligible vertices

$$U_i \coloneqq \big\{ v \in V \setminus U \mid \mathrm{rt}\big(G, (p \cup \{q_i \mapsto v\}) \circ \hat{p}^{-1}\big) \leq c_{\min} \big\} \,. \tag{18}$$

Now we try to find $v_1^* \neq v_2^*$ as $(v_1^*, v_2^*) \coloneqq \arg\min_{(v_1,v_2)\in U_1\times U_2} d(v_1, v_2)$. If such (v_1^*, v_2^*) does not exist, we do not include q_1 and q_2 in p; otherwise, we set $p \leftarrow p \cup \{q_1 \mapsto v_1^*, q_2 \mapsto v_2^*\}$ and update $U \leftarrow U \cup \{v_1^*, v_2^*\}$. After iterating over all gates in $\mathrm{tg}(L)$, we return p.

B.3 Greedy Size Mapper

The *greedy size mapper* the same as the greedy depth mapper, except that we replace $\mathrm{rt}(\cdot)$ with $\mathrm{rs}(\cdot)$ in (16).

B.4 Simple Size Mapper

The *simple size mapper* places only the lowest-cost gate at its lowest-cost location. More precisely, we find a placement $p \coloneqq \{q_1 \mapsto v_1, q_2 \mapsto v_2\}$ that attains the optimum

$$\min_{(q_1,q_2)\in \mathrm{tg}(L)} \min_{(v_1,v_2)\in E} \mathrm{rs}\big(G, (p \cup \{q_1 \mapsto v_1, q_2 \mapsto v_2\}) \circ \hat{p}^{-1}\big) \tag{19}$$

where we consider all orderings of the edges of E, and return p. Note that we have replaced $\mathrm{rt}(\cdot)$ with $\mathrm{rs}(\cdot)$ in (17).

B.5 Extension Size Mapper

The *extension size mapper* first finds an initial placement p using (19). Let $c'_{\min}$ be the value attained at the optimum for (19). After finding the initial placement, we try to only place another gate if it is cheaper to place now rather than in a later call to the mapper.

Specifically, for the current p and $\hat{p}$, we define $\hat{p}' \colon Q \to V$ as the placement after performing the permutation circuit constructed from transpositions achieving $\mathrm{rs}(G, p \circ \hat{p}^{-1})$. Let $U \coloneqq \emptyset$. Now we define a heuristic for the number of saved transpositions, $s \colon Q \times Q \to \mathbb{N}$, as

$$\begin{aligned} s(q_1,q_2) \coloneqq \mathrm{rs}\big(G, p \circ \hat{p}^{-1}\big) + \min_{(v_1,v_2)\in E} \mathrm{rs}\Big(G, \{q_1 \mapsto v_1, q_2 \mapsto v_2\} \circ (\hat{p}')^{-1}\Big) \\ - \min_{(u_1,u_2)\in E'} \mathrm{rs}\big(G, (p \cup \{q_1 \mapsto u_1, q_2 \mapsto u_2\}) \circ \hat{p}^{-1}\big)\,, \end{aligned} \tag{20}$$

where E' is the edge set of $G[V \setminus U]$ and we consider all orderings of the edges of E and E'.

The extension size mapper iterates the following. We find the gate $(q_1^*, q_2^*) \in \mathrm{tg}(L)$ attaining $s_{\max} \coloneqq \max_{(q_1,q_2)\in \mathrm{tg}(L)} s(q_1,q_2)$, and let $(u_1^*, u_2^*) \in E'$ be the edge attaining $s_{\max}$ as given by (20). If $s_{\max} \geq 0$, we set $p \leftarrow p \cup \{q_1^* \mapsto u_1^*, q_2^* \mapsto u_2^*\}$, remove the gate (q_1^*, q_2^*) from L, update $U \leftarrow U \cup \{v_1^*, v_2^*\}$, and iterate; otherwise, we stop and return p.

B.6 Qiskit-based Mapper

Finally, we implement a mapper that is based on Qiskit's circuit transformation (described in Section 2.2.1). Since this is a mapper, we only execute one iteration of the circuit transformation: for the first layer L. We also do not modify the output circuit, but instead return the final $\hat{p}$ that would be induced by executing all SWAPs found during the mapping process.

We make three changes to Qiskit's circuit transformation. The first is that when minimizing S, instead of choosing a maximal set of SWAPs in every iteration, we choose only one SWAP along an edge $e \in E$ that minimizes S. The second is that the upper bound on the number of iterations is raised to $|V|^2$, since we only apply one SWAP per iteration. Thirdly, if no trial is successful, we fall back to the simple size mapper and return the placement it finds, which places only one gate in this iteration.

C Time Complexity Analysis

To show that our proposed algorithms have polynomial worst case time complexity, we compute time complexities of our circuit transformations, permuters, and mappers explicitly.

C.1 Circuit Transformations

Greedy Swap Circuit Transformation. We ignore the initial placement since it is insignificant for large circuits. A gate from L is executed in at most $\mathrm{diam}(G)$ iterations, where $\mathrm{diam}(G)$ is the diameter of G. In every iteration, $O(|E|)$ edges are checked to determine gates that can be executed and SWAPs that will decrease R. Therefore, the total time complexity is $O(|C||E|\,\mathrm{diam}(G))$, where $|C|$ denotes the size of circuit C. There is a tighter bound in terms of output circuit C' since every iteration creates a layer in the transformed circuit, the complexity is $O(\mathrm{depth}(C')|E|)$, where $\mathrm{depth}(C')$ denotes the circuit depth of C'.

Specialized Circuit Transformations (Section 2.2.3). We again ignore the time complexity of computing the initial placement. Let t_m be an upper bound on the time complexity of the mapper, and let t_p be an upper bound on the time complexity of the permuter. Computing $p \circ \hat{p}^{-1}$ takes time $O(|V|)$. The number of transpositions produced by the permuter is at most t_p, so executing the associated SWAPs takes time $O(t_p)$. Only one gate from L may be executed every iteration so we upper bound the number of iterations by $|C|$. We find a time complexity of $O(|C|\,(t_m + |V| + t_p))$. Clearly, if $t_p, t_m \in \mathrm{poly}(|C|, |V|)$ then our circuit transformation is also poly-time as desired.

C.2 Permuters

We show that the permuters are polynomial-time in the input size.

Complete Graph. The time complexity of the ROUTING VIA MATCHINGS algorithm for K_n is $O(n)$ [2]. The other operations described above also take time $O(n)$, so we get a time complexity of $O(n)$ for the complete graph permuter.

Path Graph. Constructing the completion $\hat{\pi}$ takes time $O(|V|)$. The total complexity for running the path permuter is $O(|V|^2)$, where the time complexity of the ROUTING VIA MATCHINGS algorithm [2] dominates the construction of $\hat{\pi}$.

Hierarchical Product. Let t_1 and t_2 upper bound the time complexity of algorithms for PARTIAL ROUTING VIA MATCHINGS on G_1 and G_2, respectively. We first find $\deg(\pi)$ distinct sets of representative vertices by Lemma 3. The time to find one set of representative vertices is dominated by the time to find the maximum bipartite matching, $O(n_1^{2.5})$ [23]. Then, for $\lceil \deg(\pi)/\,\mathrm{ham}(\vec{v}) \rceil$ iterations, we route on all copies of G_2 and then G_1 in parallel. Overall, we get a time complexity of

$$O\left(\deg(\pi) \cdot n_1^{2.5} + \left\lceil \frac{\deg(\pi)}{\mathrm{ham}(\vec{v})} \right\rceil (\mathrm{ham}(\vec{v})t_1 + n_1 t_2) + n_1 t_2\right). \tag{21}$$

Modular Architecture. We evaluate the time complexity of this permuter using Equation (21). We have $t_1 = O(n_1)$ and $t_2 = O(n_2)$, giving an overall time complexity of $O(dn_1^{2.5} + n_1 n_2)$, where we noted that $t_2 = O(1)$ while doing the $\deg(\pi)$ rounds of routing.

Approximation Algorithm for Partial Token Swapping. Computing an all-to-all distance matrix takes time $\Theta(|V|^3)$ using the Floyd-Warshall algorithm [17], but this cost needs only to be incurred once for a graph so we do not include it. A happy or unhappy swap can be found in time $O(|E|)$ by finding cycles in an auxiliary directed graph [38]. Similarly, finding no-token swaps has time complexity $O(|E|)$. Therefore, we get a total time complexity of $O(S|E|) \leq O(|V|^2|E|)$.

C.3 Mappers

We give polynomially-sized upper bounds on the time complexity of the mappers as a function of the time complexity of the permuter, t_p.

Greedy Depth Mapper. We perform at most $\min\{|L|, |M|\}$ iterations to place gates. In each iteration, we find a p' according to (16) in time $O(|L||M|t_p)$. Thus, the time complexity for one call of the mapper is

$$O\Big(\min\{|L|, |M|\}\left(|L||M|t_p + \sqrt{|V|}|E|\right)\Big), \tag{22}$$

where $O(\sqrt{|V|}|E|)$ is the complexity of computing a maximum matching [37].

Incremental Depth Mapper. We get $O(|L|\left(|E|t_p + |V|t_p + |V|^2\right))$ for the time complexity of the incremental depth mapper. This assumes we have access to the all-pairs distance matrix of the architecture graph, which can be precomputed in time $\Theta(|V|^3)$ [17] (independent of the input circuit).

Simple Size Mapper. The time complexity of the simple size mapper is $O(|L||E|t_p)$.

Extension Size Mapper. Calculating $s(q_1, q_2)$ for any $q_1, q_2 \in Q$ takes time $O(|E|t_p)$. Therefore, the total time complexity of the extension size mapper is $O(|L|^2|E|t_p)$.

Qiskit-based Mapper. First, we compute an all-to-all distance matrix in time $\Theta(|V|^3)$ [17], which we ignore since it is a one-time cost dependent only on the architecture. Each of the $O(|V|^2)$ iterations has a time complexity of $O(|E||L|)$. Thus, the Qiskit mapper has time complexity $O(|V|^2|E||L|)$.

The RGB No-Signalling Game

Xavier Coiteux-Roy
Facoltà di scienze informatiche, Università della Svizzera italiana, Lugano, Switzerland
xavier.coiteux.roy@usi.ch

Claude Crépeau
School of Computer Science, McGill University, Montréal, Québec, Canada
crepeau@cs.mcgill.ca

Abstract

Introducing the simplest of all No-Signalling Games: the RGB Game where two verifiers interrogate two provers, Alice and Bob, far enough from each other that communication between them is too slow to be possible. Each prover may be independently queried one of three possible colours: Red, Green or Blue. Let a be the colour announced to Alice and b be announced to Bob. To win the game they must reply colours x (resp. y) such that $a \neq x \neq y \neq b$.

This work focuses on this new game mainly as a pedagogical tool for its simplicity but also because it triggered us to introduce a new set of definitions for reductions among multi-party probability distributions and related *non-locality classes*. We show that a particular winning strategy for the RGB Game is equivalent to the PR-Box of Popescu-Rohrlich and thus No-Signalling. Moreover, we use this example to define No-Signalling in a new useful way, as the intersection of two natural classes of multi-party probability distributions called one-way signalling. We exhibit a quantum strategy able to beat the classical local maximum winning probability of 8/9 shifting it up to 11/12. Optimality of this quantum strategy is demonstrated using the standard tool of semidefinite programming.

2012 ACM Subject Classification Theory of computation → Quantum information theory

Keywords and phrases No-Signalling, Quantum Entanglement, Non-Locality, Bell inequality, Semi-definite Programming, Non-locality Hierarchy

Digital Object Identifier 10.4230/LIPIcs.TQC.2019.4

Funding *Xavier Coiteux-Roy*: Supported in part by SNF and FRQNT.
Claude Crépeau: Supported in part by Québec's FRQNT (INTRIQ) and Canada's NSERC.

Acknowledgements We thank Gilles Brassard, Arne Hansen, Adel Magra, Alberto Montina, Louis Salvail, Stefan Wolf and Nan Yang for discussions about early versions of this work.

1 The Game

Claude started this research trying to find the simplest example he could think of to illustrate multi-party distributions achievable via entanglement and No-Signalling in general. His interest started from the following question on Quora: "Could someone explain quantum entanglement to me like I'm 5 years old?" Jon Hudson [10], a former Stanford QM student, had given an answer involving friends choosing to have pizza (or not) on the Moon and on Earth but he did not quite come up with a crisp No-Signalling situation. Claude cooked up the RGB example after reading Jon's answer.

The canonical examples in this area are the Magic Square Game [12, 13] and the so-called PR-box [14] of Popescu-Rohrlich, both of which require some basic notions of arithmetics to be introduced, or at least some basic logic as a common background. The purpose now is to present an example so simple that even a five year old would understand it!

The RGB game is as follows:

> “ Two people, Alice and Bob, play a game with friends Albert and Boris. Alice and Albert are on the moon, while Bob and Boris stay on earth. Albert and Boris each independently picks at random a colour out of three possibilities: Red, Green or Blue, and locally tells it to Alice or Bob.
> Right away Alice and Bob choose a colour different from the one provided by their local counterpart. For instance, if Albert tells Green to Alice, she may choose Red or Blue, while if Boris tells Red to Bob, he may choose Blue or Green.
> Alice and Bob win the game if they never answer the same colour, either Red-Blue, Red-Green or Blue-Green in the example above. ”

Figure 1 summarizes the input/output relation that Alice and Bob must satisfy. a is the colour given to Alice and b is the colour given to Bob. Their answers are x and y respectively. The condition they are trying to achieve is simply $a \neq x \neq y \neq b$.

Figure 1 The $\mathcal{RGB}$-box such that $a \neq x \neq y \neq b$.

Such boxes are a standard way of representing the possible behaviours of Alice and Bob. Indeed we can think of this box as a channel precisely describing the distribution of x, y given fixed values of a, b. The box of Figure 1 does not specify the probabilities exactly and thus the name of the box is in calligraphic letters representing the set of all the distributions that satisfy the given conditions. There are many distinct ways of fulfilling the conditions of the game and many distributions that will win the game 100% of the time.

1.1 Winning Strategies

Let's first consider a deterministic strategy for Alice and Bob's behaviour as described by the box of Figure 2.

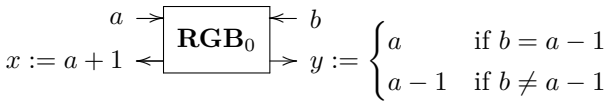


Figure 2 A deterministic $\mathbf{RGB}_0$-box.

In this example we assume the colours are labelled $0, 1$ or 2 and that arithmetic operations are performed modulo 3. When a and b are the same colour u it produces

$$a = u,\ x = u + 1,\ y = u - 1,\ b = u.$$

The values $u + 1$ and $u - 1$ are the other two colours, distinct from u. However, when a and b are the distinct colours u, v it produces either

$$a = u,\ x = u + 1,\ y = u,\ b = v$$

when the third colour is $u + 1 = v - 1$ or

$$a = u,\ x = u + 1,\ y = u - 1,\ b = v$$

when the third colour is $u - 1 = v + 1$.

This deterministic strategy defines completely the probability distribution of the outputs x, y given a, b: $\Pr(x, y|a, b)$ is zero except when $x = a + 1$ and $y = \begin{cases} a & \text{if } b = a - 1 \\ a - 1 & \text{if } b \neq a - 1 \end{cases}$ in which case it is precisely one. Therefore we name this box $\mathbf{RGB}_0$ with bold characters because it precisely defines a unique probability distribution $P_{x,y|a,b}$. This box achieves the prescribed condition $a \neq x \neq y \neq b$ in a unique deterministic way for each a, b.

After complete examination of this condition one realizes that when $a = b$ is a single colour u the conditions can be satisfied in exactly two ways

$$a = u,\ x = u \pm 1,\ y = u \mp 1,\ b = u$$

whereas when a and b are distinct colours u, v the conditions can be satisfied in exactly three ways

$$a = u,\ x = v,\ y = u,\ b = v$$

$$a = u,\ x = u \pm 1,\ y = v \pm 1,\ b = v.$$

From this we conclude that out of the 9 possible a, b pairs, three of them ($a = b$) may have two solutions and six of them ($a \neq b$) may have three solutions. This yields a total of $2^3 3^6 = 18^3 = 5832$ distinct deterministic winning strategies. The above $\mathbf{RGB}_0$ strategy is only one of these.

We can completely parametrize all the winning strategies as a function of 15 real parameters $p_0, p_1, p_2, p_{01}, p_{02}, p_{10}, p_{12}, p_{20}, p_{21}, q_{01}, q_{02}, q_{10}, q_{12}, q_{20}, q_{21}$ in the interval $[0, 1]$ such that $p_{uv} + q_{uv} \leq 1$ as follows

$$P_{u+1,u-1|u,u} = p_u \text{ and } P_{u-1,u+1|u,u} = 1 - p_u, \text{ for } u \in \{0, 1, 2\} \tag{1}$$

$$P_{w,u|u,v} = p_{uv},\ P_{v,w|u,v} = q_{uv} \text{ and } P_{v,u|u,v} = 1 - p_{uv} - q_{uv}, \text{ for } \{u, v, w\} = \{0, 1, 2\}. \tag{2}$$

All the winning strategies to this game are among these probability distributions. They are all the valid convex combinations of the 5832 distinct deterministic winning strategies.

The deterministic strategy $\mathbf{RGB}_0$ of Figure 2 is the special case

$$p_0 = p_1 = p_2 = p_{02} = p_{20} = q_{01} = q_{10} = q_{12} = q_{21} = 1$$

$$p_{01} = p_{10} = p_{12} = p_{21} = q_{02} = q_{20} = 0.$$

The rest of this paper is going to focus on exactly one of these strategies with a very remarkable property: it *does not require* Alice and Bob to signal to implement it (whereas all the others actually do). This strategy is going to be named $\mathbf{R}^{\mathbf{GR}}_{\mathbf{BG}}\mathbf{B}$[†] and is specified by the parameters

$$p_0 = p_1 = p_2 = p_{01} = p_{10} = p_{02} = p_{20} = p_{12} = p_{21} = q_{01} = q_{10} = q_{02} = q_{20} = q_{12} = q_{21} = \frac{1}{2}.$$

In Figure 3, $\mathbf{R}^{\mathbf{GR}}_{\mathbf{BG}}\mathbf{B}$ is made precise by enforcing extra conditions on top of $a \neq x \neq y \neq b$. We force $P_{v,u|u,v} = 0$ by adding $(x, y) \neq (b, a)$. Uniformity finally imposes that all the remaining non-zero probabilities be exactly $\frac{1}{2}$.

[†] The name is a reminder that this strategy has the feature that whenever a and b are distinct, $axyb$ is $abcb$ or $acab$ (c being the third colour) but never $abab$. $\mathbf{R}^{\mathbf{GR}}_{\mathbf{BG}}\mathbf{B}$ is a combined string of types $a\underline{\ \ }_{bc}b$, $a^{ca}\underline{\ \ }b$.

Figure 3 The $\mathbf{R}^{\mathbf{GR}}_{\mathbf{BG}}\mathbf{B}$-box such that $a \neq x \neq y \neq b$, and $(x, y) \neq (b, a)$, uniformly among solutions.

Figure 4 The **PR**-box satisfying the CHSH condition, that $a \wedge b = x \oplus y$, uniformly among solutions.

1.2 Our Results

The contributions of the paper are

1. Novel notion of reducibility among strategies
2. Novel definitions of basic notions such as locality, signalling, one-way signalling and no-signalling
3. A proof that our notion of no-signalling is equivalent to the generally accepted one
4. A proof of equivalence between $\mathbf{R}^{\mathbf{GR}}_{\mathbf{BG}}\mathbf{B}$ and the well-known Popescu-Rohrlich Non-Local (yet No-Signalling) **PR**-box (see Figure 4). This Implies that $\mathbf{R}^{\mathbf{GR}}_{\mathbf{BG}}\mathbf{B}$ is also complete for the set of No-Signalling (two-party) distributions
5. A proof that $\mathbf{R}^{\mathbf{GR}}_{\mathbf{BG}}\mathbf{B}$ is the ONLY No-Signalling distribution winning the RGB game
6. A deterministic (and local) strategy with winning probability 8/9
7. A proof of optimality of this local strategy
8. Quantum strategy with winning probability 11/12
9. A proof of optimality of this quantum strategy using semidefinite programming
10. Some related open problems

2 Definitions

In this section we solely focus on the two-party single-round games and strategies that are sufficient to discuss and analyze the strategies for the RGB game. Definitions and proofs for complete generalizations to multi-party multi-round games and strategies will appear in a forthcoming paper with co-authors Adel Magra and Nan Yang.

2.1 Strategies: Two-Party Channels

2.1.1 Games

Let V be a predicate on $A \times B \times X \times Y$ (for some finite sets A, B, X, and Y) and let π be a probability distribution on $A \times B$. Then V and π define a (single-round) game G as follows: A pair of questions (a, b) is randomly chosen according to distribution π, and $a \in A$ is sent to Alice and $b \in B$ is sent to Bob. Alice must respond with an answer $x \in X$ and Bob with an answer $y \in Y$. Alice and Bob win if V evaluates to 1 on (a, b, x, y) and lose otherwise.

2.1.2 Strategies

A strategy for Alice and Bob is simply a probability distribution $P_{(x,y|a,b)}$ describing exactly how they will answer (x, y) on every pair of questions (a, b). We now breakdown the set of all possible strategies for Alice and Bob according to their degree of *non-locality*.

2.1.3 Deterministic and Local Strategies

A strategy $P_{(x,y|a,b)}$ is *deterministic* if there exists functions $f_A : A \to X, f_B : B \to Y$ such that

$$P_{(x,y|a,b)} = \begin{cases} 1 & \text{if } x = f_A(a) \text{ and } y = f_B(b) \\ 0 & \text{otherwise} \end{cases}.$$

A deterministic strategy corresponds to the situation where Alice and Bob agree on their individual actions before any knowledge of the values a, b is provided to them. In this case they use only their own input to determine their individual output.

A strategy $P_{(x,y|a,b)}$ is *local* if there exists a finite set R, functions $f_A : A \times R \to X, f_B : B \times R \to Y$ and a probability distribution $\pi_r, r \in R$, such that

$$P_{(x,y|a,b)} = \sum_{r \in R : x = f_A(a,r) \text{ and } y = f_B(b,r)} \pi_r.$$

A local strategy corresponds to the situation where Alice and Bob agree on a deterministic strategy selected uniformly among $|R|$ such possibilities. The choice r of Alice and Bob's strategy, and the choice of inputs (a, b) provided to Alice and Bob are generally agreed to be statistically independent random variables.

2.2 Local Reducibility

We now turn to the notion of locally reducing a strategy to another, that is how Alice and Bob limited to local strategies but equipped with a particular (not necessarily local) strategy U' are able to achieve another particular (not necessarily local) strategy U. For this purpose we introduce a notion of distance between strategies in order to analyze strategies that are approaching each other asymptotically.

Several distances could be selected here as long as their meaning as it approaches zero are the same. In the definitions below, U, U' are strategies and $\mathcal{U}'$ is a finite set of strategies.

▶ **Definition 1.** $|U, U'| = \sum_{a,b,x,y} |P_U(x, y|a, b) - P_{U'}(x, y|a, b)|$.

▶ **Definition 2.** $|U, \mathcal{U}'| = \min_{U' \in \mathcal{U}'} |U, U'|$.

For natural integer n, we define the set $\mathrm{LOC}^n(U)$ of strategies that are local extensions (of order n) of U to be all the strategies Alice and Bob can achieve using local strategies where strategy U may be used up to n times as sub-routine calls[‡].

▶ **Definition 3.** *We say that U' Locally Reduces to U ($U' \leq_{\mathrm{LOC}} U$) iff* $\lim_{n\to\infty} |U', \mathrm{LOC}^n(U)| = 0$.

▶ **Definition 4.** *We say that U' is Locally Equivalent to U ($U' =_{\mathrm{LOC}} U$) iff $U' \leq_{\mathrm{LOC}} U$ and $U \leq_{\mathrm{LOC}} U'$.*

Note: a similar notion of reducibility has been previously defined by Dupuis, Gisin, Hasidim, Méthot, and Pilpel [8] but without taking the limit to infinity. In their model they have previously showed that n instances of the PR-box modulo p cannot be used to replicate exactly the PR-box modulo q, for any distinct primes p, q. However, Forster and Wolf [9] have previously proved that **PR** is complete for No-Signalling distributions under a similar (asymptotic) definition.

[‡] Done by selecting functions $f_A^0 : A \times R \to A$, $f_A^1 : A \times X \times R \to A, ...$, $f_A^{n-1} : A \times X^{n-1} \times R \to A$, $f_A^n : A \times X^n \times R \to X$ to determine the input of each sub-routine from input a and previous outputs.

2.3 Locality and Non-Locality

We now define the lowest of the non-locality classes $\mathbb{LOC}$. We could define it directly from the notion of local strategies as defined above, but for analogy with the other classes we later define, $\mathbb{LOC}$ is defined as all those strategies locally reducible to a *complete* strategy we call **ID** (see Figure 5) for obvious reasons. Of course, any strategy is complete for this class.

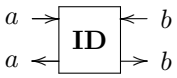


Figure 5 The **ID**-box.

▶ **Definition 5.** $\mathbb{LOC} = \{U | U \leq_{\text{LOC}} \mathbf{ID}\}$.

Note: $\mathbb{LOC}$ is the class of strategies that John Bell [4] considered as classical hidden-variable theories and that he opposed to entanglement. It is also the class of strategies that BenOr, Goldwasser, Kilian and Wigderson [5] chose to define classical Provers in Multi-Provers Interactive Proof Systems.

2.4 One-Way Signalling

We now turn to One-Way Signalling which allows communication from one side to the other. We name the directions arbitrarily Left and Right. We define **R**-$\mathbb{SIG}$ (resp. **L**-$\mathbb{SIG}$) as all those strategies locally reducible to a *complete* strategy we call **R-SIG** (see Figure 6) (resp. **L-SIG** (see Figure 7)). These classes are useful to define what it means for a strategy to *signal* as well as the notion of *No-Signalling* strategies.

Figure 6 The **R-SIG**-box.

▶ **Definition 6.** **R**-$\mathbb{SIG} = \{U | U \leq_{\text{LOC}} \mathbf{R\text{-}SIG}\}$.

▶ **Definition 7.** *We say that U Right Signals (is* **R**-$\mathbb{SIG}$*-verbose*[§]*) iff* $\mathbf{R\text{-}SIG} \leq_{\text{LOC}} U$.

Figure 7 The **L-SIG**-box.

▶ **Definition 8.** **L**-$\mathbb{SIG} = \{U | U \leq_{\text{LOC}} \mathbf{L\text{-}SIG}\}$.

▶ **Definition 9.** *We say that U Left Signals (is* **L**-$\mathbb{SIG}$*-verbose) iff* $\mathbf{L\text{-}SIG} \leq_{\text{LOC}} U$.

▶ **Definition 10.** *We say that U Signals iff U Right Signals or Left Signals.*

[§] We define the notion of $\mathbb{L}$-verbose in analogy to $\mathbb{NP}$-hard: it means "as verbose as any distribution in non-locality class $\mathbb{L}$". In consequence, a distribution U is $\mathbb{L}$-complete if $U \in \mathbb{L}$ and U is $\mathbb{L}$-verbose.

We prove a first result that is intuitively obvious. We show that the complete strategy **R-SIG** cannot be approximated in **L**-$\mathbb{SIG}$ and the other way around.

▶ **Theorem 11.** **R-SIG** $\notin$ **L**-$\mathbb{SIG}$ *and* **L-SIG** $\notin$ **R**-$\mathbb{SIG}$.

Proof. Follows from a simple capacity argument. For all n, all the channels in LOC^n(**R-SIG**) have zero left-capacity, while **L-SIG** has non-zero left-capacity. And vice-versa. ◀

2.5 Signalling

We are now ready to define the largest of the non-locality classes named $\mathbb{SIG}$. Indeed every possible strategy is in $\mathbb{SIG}$.

▶ **Definition 12.** $\mathbb{SIG} = \{U | U \leq_{\mathrm{LOC}} \mathbf{SIG}\}$.

Figure 8 The **SIG**-box.

▶ **Definition 13.** *We say that U Fully Signals (is $\mathbb{SIG}$-verbose) iff U Right Signals and Left Signals.*

2.6 No-Signalling

We finally define the less intuitive non-locality class $\mathbb{NOSIG}$ in relation to classes defined above.

▶ **Definition 14.** $\mathbb{NOSIG} = \mathbf{R}\text{-}\mathbb{SIG} \bigcap \mathbf{L}\text{-}\mathbb{SIG}$.

A similar characterization may be found in [1] Section 3 and [2] Corollary 3.5.

▶ **Theorem 15.** *The above definition of $\mathbb{NOSIG}$ exactly coincides with the* traditional *notion of No-Signalling [3].*

Proof. If U is signalling then it is verbose for at least one of **R**-$\mathbb{SIG}$ or **L**-$\mathbb{SIG}$. Without loss of generality, assume it is verbose for **R**-$\mathbb{SIG}$. Then by theorem 11, $U \notin$ **L**-$\mathbb{SIG}$, thus $U \notin \mathbf{R}\text{-}\mathbb{SIG} \bigcap \mathbf{L}\text{-}\mathbb{SIG}$.

If U is no-signalling then Alice's marginal distribution is independent from Bob's input b. Therefore, she can sample an output x according to her input a only as $P_{X|A=a}$ deduced from $P_{X,Y|A,B}$. Alice can now communicate a, x to Bob. Bob given a, b, x can select y according to the distribution $P_{Y|A=a,B=b,X=x}$ deduced from $P_{X,Y|A,B}$. The produced x, y will have distribution $P_{X,Y|A=a,B=b}$ as expected. This proves $U \in$ **R**-$\mathbb{SIG}$. Membership to **L**-$\mathbb{SIG}$ is proven similarly. ◀

Figure 9 shows the relation of these classes as well as the case obtained via quantum entanglement ($|\mathbb{LOC}\rangle$) as considered by Bell [4] and via commuting-operators ($\mathbb{COMOP}$) as defined by Ito, Kobayashi, Preda, Sun, and Yao [11].

▶ **Definition 16.** *We say that U does not Signal iff U does not Right Signal nor Left Signal iff $U \in \mathbb{NOSIG}$.*

▶ **Theorem 17.** *If $U \in$ **R**-$\mathbb{SIG}$ (or $U \in$ **L**-$\mathbb{SIG}$) and U is symmetric then U does not Signal.*

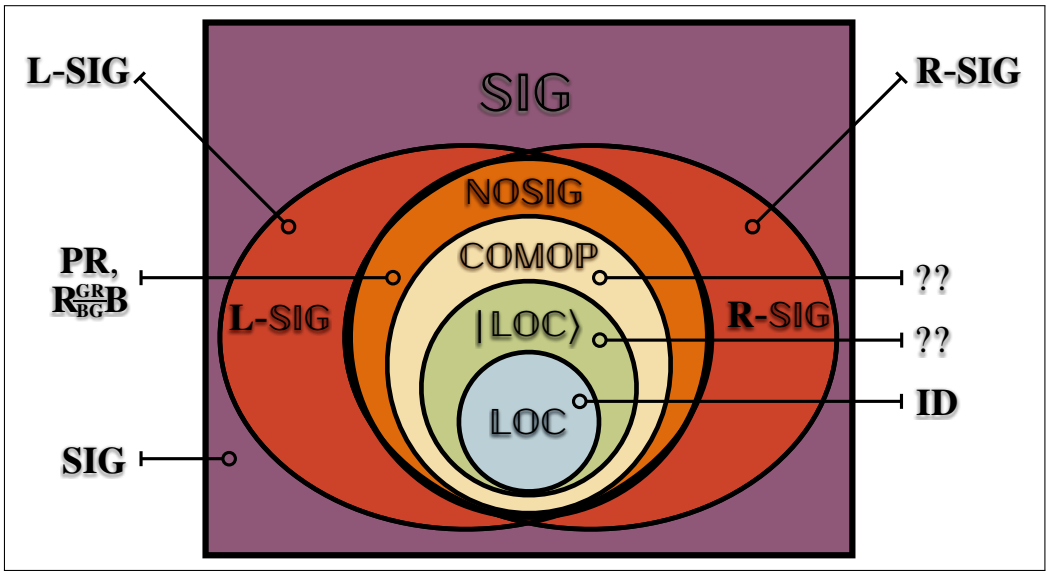


Figure 9 Non-locality Hierarchy and complete (two-party) distributions in certain classes.

Proof. $U \in \mathbf{R}\text{-}\mathbb{SIG}$ and U is symmetric imply that $U \in \mathbf{L}\text{-}\mathbb{SIG}$ as well. Thus $U \in \mathbf{R}\text{-}\mathbb{SIG} \bigcap \mathbf{L}\text{-}\mathbb{SIG}$. ◀

▶ **Theorem 18.** $\mathbf{R}_{\mathbf{BG}}^{\mathbf{GR}}\mathbf{B} \in \mathbb{NOSIG}$.

Proof. $\mathbf{R}_{\mathbf{BG}}^{\mathbf{GR}}\mathbf{B} \in \mathbf{R}\text{-}\mathbb{SIG}$ and $\mathbf{R}_{\mathbf{BG}}^{\mathbf{GR}}\mathbf{B}$ is symmetric. ◀

▶ **Theorem 19.** $\mathbf{R}_{\mathbf{BG}}^{\mathbf{GR}}\mathbf{B} =_{\mathrm{LOC}} \mathbf{PR}$.

Proof. First we show how $\mathbf{PR}$ may be achieved from $\mathbf{R}_{\mathbf{BG}}^{\mathbf{GR}}\mathbf{B}$, more precisely that $\mathbf{PR} \in \mathrm{LOC}^1(\mathbf{R}_{\mathbf{BG}}^{\mathbf{GR}}\mathbf{B})$. All arithmetic operations are performed modulo 3. Let $a' := f_A^1(a) := a$, and $b' := f_B^1(b) := 2b$. The possible pairs for (a', b') are therefore $(0,0), (0,2), (1,0), (1,2)$. Let $(x', y') \leftarrow \mathbf{R}_{\mathbf{BG}}^{\mathbf{GR}}\mathbf{B}(a', b')$. Let $x := f_A^2(a, x') := 2(x' - a + 1)$, and $y := f_B^2(b, y') := 2(y' - 2b + 1)$. We leave it as an exercice to check that (x, y) indeed satisfy the CHSH condition that $x \oplus y = a \wedge b$.

Secondly, we show how $\mathbf{R}_{\mathbf{BG}}^{\mathbf{GR}}\mathbf{B}$ may be achieved from $\mathbf{PR}$, more precisely that $\mathbf{R}_{\mathbf{BG}}^{\mathbf{GR}}\mathbf{B} \in \mathrm{LOC}^2(\mathbf{PR})$. Again, all arithmetic operations are performed modulo 3[¶]. The intuition in this case is that if $a = b$ then (x, y) should be either $(a+1, b-1)$ or $(a-1, b+1)$ at random. If $a \neq b$ then (x, y) should be either $(a+1, b+1)$ or $(a-1, b-1)$ at random. The following computations achieve precisely this using the identity $a = b$ iff $(\neg a' \oplus b') \wedge (\neg a'' \oplus b'')$, where a primed variable is the corresponding most significant bit and a double-primed variable is the corresponding least significant bit.

The first pair of functions compute the negation of the most significant bit of their inputs: let $a' := f_A^1(a) := 1 - 2(a-1)a$, and $b' := f_B^1(b) := 1 - 2(b-1)b$. Let $(x', y') \leftarrow \mathbf{PR}(a', b')$.

The second pair of functions compute the negation of the least significant bit of their inputs: let $a'' := f_A^2(a, x') := 1 - 2(a-1)(a+1)$, and $b'' := f_B^2(b, y') := 1 - 2(b-1)(b+1)$. Let $(x'', y'') \leftarrow \mathbf{PR}(a'', b'')$.

The third pair of functions compute $a \pm 1, b \pm 1$ according to the intuitive rule above: let $x := f_A^3(a, x', x'') := a + 2^{a_1 * a_2 + x' + x''}$, and $y := f_B^3(b, y', y'') := b + 2^{b_1 * b_2 + y' + y''}$. ◀

[¶] Therefore modulo 2 for the exponents according to Fermat's little theorem.

▶ **Corollary 20.** $\mathbf{R}^{\mathbf{GR}}_{\mathbf{BG}}\mathbf{B}$ *is* $\mathbb{NOSIG}$*-Complete.*

Proof. Since **PR** was previously proved $\mathbb{NOSIG}$-Complete by Forster and Wolf [9] , then so is $\mathbf{R}^{\mathbf{GR}}_{\mathbf{BG}}\mathbf{B}$. ◀

▶ **Theorem 21.** $\mathbf{R}^{\mathbf{GR}}_{\mathbf{BG}}\mathbf{B}$ *is the ONLY strategy winning the RGB game that is also No-Signalling.*

Proof. Using the notation of Equations (1) – (2), for No-Signalling on Alice's side we need for all $0 \leq u \leq 2$

$$P_{u+1,u-1|u,u} = p_u = P_{u+1,u-1|u,u+1} + P_{u+1,u|u,u+1} = 1 - p_{u\,u+1} = P_{u+1,u|u,u-1} = p_{u\,u-1}$$

and symmetrically on Bob's side

$$P_{u-1,u+1|u,u} = 1-p_u = P_{u-1,u+1|u+1,u} + P_{u,u+1|u+1,u} = 1-q_{u+1\,u} = P_{u,u+1|u-1,u} = q_{u-1\,u}.$$

Using all 6 sets of equalities we can get rid of all the variables but p_0, p_1, p_2 by setting

$$p_{u\,u-1} = q_{u+1\,u} = p_u \text{ and } p_{u\,u+1} = q_{u-1\,u} = 1 - p_u, 0 \leq u \leq 2.$$

It follows that

$$P_{u+1,u|u,u+1} = p_u + p_{u+1} - 1 = -P_{u,u+1|u+1,u}, 0 \leq u \leq 2$$

and since both $P_{u+1,u|u,u+1}$ and $P_{u,u+1|u+1,u}$ must be greater or equal to zero we conclude

$$P_{u+1,u|u,u+1} = P_{u,u+1|u+1,u} = 0 \text{ and } p_u = 1 - p_{u+1}, 0 \leq u \leq 2.$$

It results that $p_0 = 1 - p_1 = p_2 = 1 - p_0 = p_1 = 1 - p_2$ and thus

$$p_0 = p_1 = p_2 = p_{01} = p_{10} = p_{12} = p_{21} = p_{20} = p_{02} = q_{01} = q_{10} = q_{12} = q_{21} = q_{20} = q_{02} = \frac{1}{2}$$

is the only solution as claimed. ◀

▶ **Theorem 22.** *The maximum local winning probability* $p^{\text{win}}_{\text{local}}$ *to the RGB game is 8/9.*

Proof. Consider $f(R) = B$ and $f(G) = f(B) = R$ as well as $g(R) = g(B) = G$ and $g(G) = B$. By inspection of these functions we conclude $p^{\text{win}}_{\text{deterministic}} \geq 8/9$ because for all inputs a, b we have $f(a) \neq a$ and $g(b) \neq b$ and 8 out of 9 input pairs (a, b) are such that $f(a) \neq g(b)$. Since it is a well known fact that $p^{\text{win}}_{\text{local}} = p^{\text{win}}_{\text{deterministic}}$, it suffices to show that $p^{\text{win}}_{\text{deterministic}} \leq 8/9$ as well.

To prove this, consider any pair of functions f, g. To obtain $f(a) \neq a$ for all a, the image of f must contain at least 2 colours. Similarly for the image of g. Since both f and g can only take 3 values, their images must have a common colour. Therefore, there exists an a and a b such that $f(a) = g(b)$. We conclude $p^{\text{win}}_{\text{deterministic}} \leq 8/9$, and therefore $p^{\text{win}}_{\text{local}} = p^{\text{win}}_{\text{deterministic}} = 8/9$. ◀

Note: somewhat surprisingly Theorem 19 is not good enough to surpass $p^{\text{win}}_{\text{local}}$ in the quantum case. Since $\mathbf{R}^{\mathbf{GR}}_{\mathbf{BG}}\mathbf{B} \in \text{LOC}^2(\mathbf{PR})$ (and not in $\text{LOC}^1(\mathbf{PR})$), an optimal quantum approximation to a PR-box (known to succeed with probability $\frac{2+\sqrt{2}}{4}$) used instead of the perfect one only yields a $\frac{3}{4}$ approximation to an RGB-box.

A natural question is therefore to find a quantum strategy that is better than the local one.

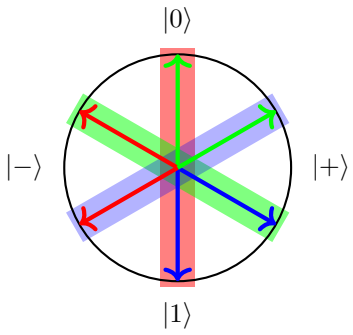


Figure 10 Alice and Bob's best quantum strategy is to each make the above projective measurement on their half-singlet. The basis (rectangle) depends on their own input colour. Their output is the colour of the measured arrow.

3 A Better-than-Local Quantum Strategy

There is indeed a better-than-local quantum strategy which wins with probability 11/12:

Alice and Bob share a singlet state $|\psi^-\rangle_{AB}$. According to their own input colour, they choose their measurement from the following list:

$$\Pi_{\text{Red}} = |0\rangle\langle 0|\,, \Pi_{\text{Green}} = \left|v^+\right\rangle\!\left\langle v^+\right|\,, \Pi_{\text{Blue}} = \left|v^-\right\rangle\!\left\langle v^-\right|\,, \tag{3}$$

where

$$\left|v^\pm\right\rangle = \frac{1}{2}|0\rangle \pm \frac{\sqrt{3}}{2}|1\rangle\,. \tag{4}$$

These 3 projectors are located in the same plane equidistantly (like the Mercedes-Benz logo). The colour names can be permutated freely as long as Alice and Bob do the same projection for the same colour.

If the output of their measurement is positive, they output the colour that comes after their input colour in the cycle RGB. Otherwise, they output the previous colour. They never output their own input colour as it leads to a sure loss.

For example, if Alice's input is Green and she measures a positive result when applying the projector Π_{Green}, then $a = G$ and $x = G + 1 = B$ (the colour addition is modulo 3). Figure 10 explains the protocol graphically.

3.1 Proof of Winning Probability

We look at the probability of losing as it is simpler. To simplify notation, we call directly $x = a-1 \leftrightarrow x = 0$ and $x = a+1 \leftrightarrow x = 1$ as well as $y = b-1 \leftrightarrow y = 0$ and $y = b+1 \leftrightarrow y = 1$. Alice and Bob lose in the following cases:

$$\begin{array}{ll} x = y & \text{if } b = a\,, \\ x = 0 \wedge y = 1 & \text{if } b = a+1 \mod 3\,, \\ x = 1 \wedge y = 0 & \text{if } b = a-1 \mod 3\,. \end{array} \tag{losing cases}$$

The probability of error E only depends on the relation between a and b and is given by

$$E_{a=b} = \mathrm{tr}\left(\left|\psi^-\right\rangle\!\left\langle\psi^-\right|_{AB} \cdot \left((\Pi_a \otimes \Pi_b) + (\Pi_a^\perp \otimes \Pi_b^\perp)\right)\right) = 0\,, \tag{5}$$

$$E_{a+1=b} = \mathrm{tr}\left(\left|\psi^-\right\rangle\!\left\langle\psi^-\right|_{AB} \cdot (\Pi_a^\perp \otimes \Pi_b)\right) = \frac{1}{8}\,, \tag{6}$$

$$E_{a-1=b} = \mathrm{tr}\left(\left|\psi^-\right\rangle\!\left\langle\psi^-\right|_{AB} \cdot (\Pi_a \otimes \Pi_b^\perp)\right) = \frac{1}{8}\,. \tag{7}$$

And the winning probability of this quantum strategy is (with uniformly random inputs):

$$p(\text{win}) = 1 - \frac{3E_{a=b} + 3E_{a+1=b} + 3E_{a-1=b}}{9} = \frac{11}{12}\,. \tag{8}$$

The game is therefore won with probability 11/12 using this quantum strategy. ◀

4 The Bell Inequality Associated to the RGB Game

The above quantum strategy is optimal among quantum strategies. To prove it in Section 5, we now analyze a Bell inequality associated to the RGB game. Bell game and Bell inequalities are equivalent formulations of the same phenomenon. We quickly recall how to translate from one paradigm to the other before defining the inequality and stating the corresponding bounds for quantum and No-Signalling strategies.

4.1 Bell Game vs Bell Inequality Notations

Up to now, we have analyzed the RGB game in the modern game context, meaning we treated strategies as probability distributions of the form $P_{x,y|a,b}$ and showed strategies in different non-locality classes (*i.e.*, local, quantum or No-Signalling) can achieve different win rates. To finetune our analysis, we excluded without losing generality the output colour that always lose (*i.e.*, $x = a$ and $y = b$) and treated the remaining outputs as binary (*i.e.*, $0 := u - 1$ and $1 := u + 1$). In the next subsections, we will use the notation $p_{(x,y|a,b)}$ for the individual conditional probabilities.

However, another way to see this problem is through Bell inequalities. Instead of looking at a game with binary outputs, one consider the properties of observables with values in $\{-1, 1\}$. An observable is simply a physical quantity one can decide to measure. In physics, Bell inequalities (*e.g.*, the CHSH inequality) are usually specified by a function of the expected correlations of different observables. This function defines a quantity to which classical mechanics (*i.e.*, local hidden-variable models) imposes a limit that can be broken using quantum mechanics. We remark that all of Alice's observables need to commute (meaning the order in which they are measured don't affect their results) with all of Bob's observable to respect the No-Signalling condition shared by $\mathbb{LOC}$, $|\mathbb{LOC}\rangle$ and $\mathbb{NOSIG}$.

The canonical example of a Bell inequality is the CHSH inequality. This Bell inequality also has a quantum limit: it is Tsirelson's bound. As we are about to see, there exists a similar bound for the RGB Bell-inequality.

The relevant point is that one can translate between the two formulations by expressing the conditional probabilities of the Bell game paradigm as expectancies of correlations in the Bell-inequality paradigm, and *vice versa*. We will in fact only need the following conversion equation:

$$p_{(x=y|a,b)} = \frac{1 + \langle A_a B_b \rangle}{2}\,, \tag{9}$$

where we noted $\langle A_a B_b \rangle$ the expected correlation between the measurement outcomes of Alice's observable A_a and Bob's observable B_b.

4.2 Intermediate Step: Rewriting the Probability of Winning as a Function of Expected Correlations

The following lemma will make the subsequent Bell-inequality formulation simple.

▶ **Lemma 23.** *The probability of a given strategy distribution winning the game is given by*

$$p^{\text{win}} = \frac{1}{9}\sum_{u=0}^{2} 2 - p_{(x=y|u,u)} + \frac{p_{(x=y|u,u+1)}}{2} + \frac{p_{(x=y|u,u-1)}}{2}\,. \tag{10}$$

It depends only on the correlations between Alice's and Bob's outputs, not on their marginals.

Proof. By looking at the three losing cases above (see Section 3), we obtain the probability of a distribution winning the game:

$$p^{\text{win}} = \frac{1}{9}\sum_{u=0}^{2} \left(3 - p_{(0,0|u,u)} - p_{(1,1|u,u)} - p_{(0,1|u,u+1)} - p_{(1,0|u,u-1)}\right)\,.$$

(winning probability equation)

We rewrite it in terms of the marginals and correlations $\{p_{(x=0|a)}, p_{(y=0|b)}, p_{(x=y|a,b)}\}$. Here is how we can transform each term:

$$p_{(0,0|a,b)} = \frac{p_{(x=0|a)} + p_{(y=0|b)} + p_{(x=y|a,b)} - 1}{2}\,, \tag{11}$$

$$p_{(1,1|a,b)} = p_{(x=y|a,b)} - p_{(0,0|a,b)}\,, \tag{12}$$

$$p_{(0,1|a,b)} = p_{(x=0|a)} - p_{(0,0|a,b)}\,, \tag{13}$$

$$p_{(1,0|a,b)} = p_{(y=0|b)} - p_{(0,0|a,b)}\,. \tag{14}$$

Replacing them into the winning probability equation gives

$$p^{\text{win}} = \frac{1}{9}\sum_{u=0}^{2} 3 - p_{(0,0|u,u)} - p_{(1,1|u,u)} - p_{(0,1|u,u+1)} - p_{(1,0|u,u-1)} \tag{15}$$

$$= \frac{1}{9}\sum_{u=0}^{2} 3 - p_{(0,0|u,u)} - p_{(x=y|u,u)} + p_{(0,0|u,u)} - p_{(x=0|a=u)} + p_{(0,0|u,u+1)} - p_{(y=0|b=u-1)} + p_{(0,0|u,u-1)} \tag{16}$$

$$= \frac{1}{9}\sum_{u=1}^{2} 3 - p_{(x=y|u,u)} - p_{(x=0|a=u)} + \frac{p_{(x=0|a=u)} + p_{(y=0|b=u+1)} + p_{(x=y|u,u+1)} - 1}{2} - p_{(y=0|b=u-1)} + \frac{p_{(x=0|a=u)} + p_{(y=0|b=u-1)} + p_{(x=y|u,u-1)} - 1}{2} \tag{17}$$

$$= \frac{1}{9}\sum_{u=0}^{2} 2 - p_{(x=y|u,u)} + \frac{p_{(x=y|u,u+1)}}{2} + \frac{p_{(x=y|u,u-1)}}{2}\,. \qquad \blacktriangleleft$$

4.3 The RGB Bell-Inequality

We show a new simple case of a Bell inequality which we call the RGB Bell-inequality. We define it by reformulating the bound on the local winning probability of the RGB game.

▶ **Proposition 24.** *The following quantity is related to the RGB game:*

$$R := \left| \sum_{i=0}^{2} -2 \langle A_i B_i \rangle + \langle A_i B_{i+1} \rangle + \langle A_i B_{i-1} \rangle \right| . \qquad \text{(RGB Bell-quantity)}$$

and allows us to express the RGB Bell-inequality:

$$R_{\text{local}} \leq 8 . \qquad \text{(RGB Bell-inequality)}$$

Proof. We first rewrite the equation describing the probability of winning the RGB game into a Bell-inequality notation by taking Lemma 23 and making the simple substitution given in Eq. 9. We obtain

$$p^{\text{win}} = \frac{1}{36} \sum_{i=0}^{2} 8 - 2 \langle A_i B_i \rangle + \langle A_i B_{i+1} \rangle + \langle A_i B_{i-1} \rangle . \tag{18}$$

We then define the interesting part as the RGB Bell-inequality:

$$R := 36 \cdot p^{\text{win}} - 24 = \left| \sum_{i=0}^{2} -2 \langle A_i B_i \rangle + \langle A_i B_{i+1} \rangle + \langle A_i B_{i-1} \rangle \right| . \tag{19}$$

Finally, from Theorem 22 we have $p^{\text{win}}_{\text{local}} \leq \frac{8}{9}$, which by the last equation implies $R_{\text{local}} \leq 8$. ◀

As we showed in Section 3, quantum mechanics allows for better-than-local strategies, but we will soon show that there is also a limit to how good quantum strategies can be. In fact, the quantum strategy we described earlier is optimal.

▶ **Theorem 25.** *The RGB Bell-inequality can be broken by quantum distributions, but there exists for the RGB game an analogue to Tsirelson's bound.*

$$R_{\text{quantum}} \leq 9 . \qquad \text{(quantum bound)}$$

The inequality is tight.

Proof. The value $R_{\text{quantum}} = 9$ is possible. It follows directly from the quantum strategy achieving a win rate of $\frac{11}{12}$ (as described in Section 3.) The proof one cannot do better is shown next in Section 5. ◀

While quantum strategies cannot reach the trivial upper bound, No-Signalling strategies can.

▶ **Proposition 26.** *No-Signalling physics (*i.e.*, access to* $\mathbf{R}^{\mathbf{GR}}_{\mathbf{BG}}\mathbf{B}$*) could break maximally the RGB Bell-inequality.*

$$R_{\text{No-Signalling}} \leq 12 . \qquad \text{(trivial No-Signalling bound)}$$

The inequality is tight.

Proof. The value $R_{\text{No-Signalling}} = 12$ is possible by using the No-Signalling strategy described in Section 1 because it achieves a win rate of 1. The inequality is tight as all expected correlation terms are here bounded by $\{-1, 1\}$. ◀

5 Tsirelson's-like Bound and Proof of Optimality of the Quantum Strategy

We now prove the optimality of the quantum strategy described in Section 3 by finding a Tsirelson's-like bound for the RGB Bell-inequality.

5.1 The Optimization Problem

We want to prove that for any $|\psi\rangle$, any $\{A_a\}$ and any $\{B_b\}$, the quantum limit for the RGB Bell-inequality holds:

$$R_{\text{quantum}} = \left| \sum_{u=0}^{2} -2 \langle A_u B_u \rangle + \langle A_u B_{u+1} \rangle + \langle A_u B_{u-1} \rangle \right| \leq 9 \,. \qquad \text{(quantum bound)}$$

We call the value associated to our known quantum strategy $R' = 9$ and the optimal value R^*.

5.2 Solving the Bell Inequality Using Semidefinite Programming

We closely follow Wehner's semidefinite programming technique [15]. The idea is first to transform the Bell-inequality problem from the quantum realm to the real-vector space using a result by Tsirelson. Then we use semidefinite programming with Lagrangian duality. The key point is that the Lagrangian dual problem upper bounds the primal problem. So by guessing a solution to the dual problem which have the same value as R', we prove that R' is optimal.

5.3 A Bell Inequality as a Real Vector Problem

We will use an important theorem by Tsirelson[‖] [7].

▶ **Theorem 27** (Tsirelson). *Let $A_1, \ldots, A_n$ and $B_1, \ldots, B_n$ be observables with eigenvalues in the interval $\{-1, 1\}$. Then for any state $|\psi\rangle \in \mathcal{A} \otimes \mathcal{B}$, there exist real unit vectors $\vec{x}_1, \ldots, \vec{x}_n, \vec{y}_1, \ldots, \vec{y}_n \in \mathbb{R}^{2n}$ such that for all $s, t \in \{1, \ldots, n\}$:*

$$\langle \psi | \, A_s \otimes B_t \, | \psi \rangle = \vec{x}_s \cdot \vec{y}_t \,. \tag{20}$$

Conversely, let $\vec{x}_s, \vec{y}_t \in \mathbb{R}^N$ be real unit vectors. Let $|\psi\rangle \in \mathcal{A} \otimes \mathcal{B}$ be any maximally entangled state where $\dim(\mathcal{A}) = \dim(\mathcal{B}) = 2^{\lceil N/2 \rceil}$. Then there exist observables A_s on $\mathcal{A}$ and B_t on $\mathcal{B}$ with eigenvalues ± 1 such that for all $s, t \in \{1, \ldots, n\}$:

$$\vec{x}_s \cdot \vec{y}_t = \langle \psi | \, A_s \otimes B_t \, | \psi \rangle \,. \tag{21}$$

Applying it to our case, we reduce our Bell-inequality problem to maximizing the following real-vectorial expression:

$$R = \sum_{i=0}^{2} -2\vec{x}_i \cdot \vec{y}_i + \vec{x}_i \cdot \vec{y}_{i+1} + \vec{x}_i \cdot \vec{y}_{i-1} \tag{22}$$

under the constraints $\forall i, ||\vec{x}_i|| = ||\vec{y}_i|| = 1$.

[‖] We write it as formulated in [15], but fix a small mistake in the quantifiers order (it was correct in the original paper).

Proof of Quantum Optimality

5.4 The Primal Problem

We re-write the last statements in a matrix form.

$$G = \begin{pmatrix} \vec{x}_1 \\ \vec{x}_2 \\ \vec{x}_3 \\ \vec{y}_1 \\ \vec{y}_2 \\ \vec{y}_3 \end{pmatrix} \cdot \begin{pmatrix} \vec{x}_1 & \vec{x}_2 & \vec{x}_3 & \vec{y}_1 & \vec{y}_2 & \vec{y}_3 \end{pmatrix} . \tag{23}$$

We note G can have this form if and only if it is semidefinite positive and that its diagonal elements are equal to 1 because of the normalization constraints. We also define the matrix W in a way that $\frac{1}{2}\operatorname{tr} GW = R_G$ where R_G is the R defined in Eq. 22 associated to this strategy G.

$$W = \begin{pmatrix} 0 & 0 & 0 & -2 & 1 & 1 \\ 0 & 0 & 0 & 1 & -2 & 1 \\ 0 & 0 & 0 & 1 & 1 & -2 \\ -2 & 1 & 1 & 0 & 0 & 0 \\ 1 & -2 & 1 & 0 & 0 & 0 \\ 1 & 1 & -2 & 0 & 0 & 0 \end{pmatrix} . \tag{24}$$

Then the semidefinite optimization primal problem is

$$\text{maximize } \frac{1}{2}\operatorname{tr} GW \quad \text{subject to} \quad G \geq 0 \text{ and } \forall i, g_{ii} = 1 . \tag{primal problem}$$

5.4.1 The Primal Solution

The quantum strategy we found previously is associated with the value $R' = 9$. For the sake of completeness, we prove again here this value is achievable.

$$G' = \begin{pmatrix} 1 & -\frac{1}{2} & -\frac{1}{2} & -1 & \frac{1}{2} & \frac{1}{2} \\ -\frac{1}{2} & 1 & -\frac{1}{2} & \frac{1}{2} & -1 & \frac{1}{2} \\ -\frac{1}{2} & -\frac{1}{2} & 1 & \frac{1}{2} & \frac{1}{2} & -1 \\ -1 & \frac{1}{2} & \frac{1}{2} & 1 & -\frac{1}{2} & -\frac{1}{2} \\ \frac{1}{2} & -1 & \frac{1}{2} & -\frac{1}{2} & 1 & -\frac{1}{2} \\ \frac{1}{2} & \frac{1}{2} & -1 & -\frac{1}{2} & -\frac{1}{2} & 1 \end{pmatrix} . \tag{primal solution}$$

We check that $G' \geq 0$ by looking at its eigenvalues: they are indeed $\{3, 3, 0, 0, 0, 0\}$. G' is therefore a feasible solution whose primal value is 9.

5.5 The Dual Problem

We now turn to the dual problem with Lagrange multipliers. The idea is to pose an objective function $\mathcal{L}(G, \Lambda)$ which will be equal to R_G if G is a feasible solution (*i.e.*, G is semidefinite positive and all the normalization constraints are satisfied) and whose dual can be evaluated in a non-trivial way.

$$\mathcal{L}(G, \Lambda) = \frac{1}{2}\operatorname{tr} GW - \operatorname{tr} \Lambda(G - I_6) , \tag{objective function}$$

where Λ is the diagonal matrix of Lagrange multipliers $\{\lambda_1, \ldots, \lambda_6\}$. Note that $\mathcal{L}(G, \Lambda) = R_G$ for a valid solution because when the constraints are satistifed: $G - I_6 = \hat{0}$.

We can associate a dual function to the objective function:

$$\lambda(\Lambda) = \max_{G \text{ is feasible}} \mathcal{L}(G, \Lambda) = \max_{G \text{ is feasible}} \operatorname{tr} G(\frac{1}{2}W - \Lambda) + \operatorname{tr} \Lambda \,. \qquad \text{(dual function)}$$

The crucial fact about this dual function $\lambda(\Lambda)$ is that it upper bounds $\mathcal{L}(G, \Lambda)$, so for any feasible quantum strategy it also upper bounds R_G (and therefore R^*). This is because [6]:

$$\lambda(\Lambda) = \max_{G \text{ is feasible}} \mathcal{L}(G, \Lambda) \geq \mathcal{L}(G^*, \Lambda) = \mathcal{L}(G^*) = R^* \,. \qquad (25)$$

5.6 The Dual Solution

We simply exhibit one matrix Λ such that this upper bound $\lambda(\Lambda)$ is 9. Since we can reach it, then it will be tight.

We observe that $\lambda(\Lambda)$ evaluates to infinity if $-\frac{1}{2}W + \Lambda \ngeq 0$, and that otherwise, the G maximizing $\mathcal{L}(G, \Lambda)$ is the null matrix. This leads to the following dual problem:

$$\text{minimize} \operatorname{tr} \Lambda \quad \text{subject to} \quad -\frac{1}{2}W + \Lambda \geq 0 \,. \qquad \text{(dual problem)}$$

We try the solution

$$\Lambda' = \frac{3}{2} I_6 \,. \qquad \text{(dual solution)}$$

The eigenvalues of $-\frac{1}{2}W + \Lambda'$ are $\{3, 3, \frac{3}{2}, \frac{3}{2}, 0, 0\}$, confirming it is semidefinite positive and thus a feasible solution (it does not lead to the trivial bound). The associated dual value is 9 and confirms the optimality of our quantum solution.

6 Conclusion and Open Questions

We have defined a new game, the RGB Game, that is very simple and there exists a No-Signalling strategy winning it with probability one. In the sense we have defined, this strategy is equivalent to the winning strategy to the PR game. We showed the RGB game can be won with probabilities

$$p^{\text{win}}_{\text{local}} = \frac{8}{9} \,, \qquad p^{\text{win}}_{\text{quantum}} = \frac{11}{12} \,, \qquad p^{\text{win}}_{\text{No-Signalling}} = 1 \,.$$

Our main open question is whether there exist $|\mathbb{LOC}\rangle$-complete and $\mathbb{COMOP}$-complete distributions. Another is to generalize our work to distributions over more than two parties.

References

1 Antonio Acín, Tobias Fritz, Anthony Leverrier, and Ana Belén Sainz. A Combinatorial Approach to Nonlocality and Contextuality. *Communications in Mathematical Physics*, 334(2):533–628, March 2015. doi:10.1007/s00220-014-2260-1.

2 Howard Barnum, Christopher A. Fuchs, Joseph M. Renes, and Alexander Wilce. Influence-free states on compound quantum systems. *CoRR*, quant-ph/0507108v1, 2005. arXiv:1504.00943.

3 Jonathan Barrett, Noah Linden, Serge Massar, Stefano Pironio, Sandu Popescu, and David Roberts. Nonlocal correlations as an information-theoretic resource. *Phys. Rev. A*, 71:022101, February 2005. doi:10.1103/PhysRevA.71.022101.

4 John S. Bell. On the Einstein-Podolsky-Rosen paradox. *Physics*, 1:195–200, 1964.

5 Michael Ben-Or, Shafi Goldwasser, Joe Kilian, and Avi Wigderson. Multi-prover Interactive Proofs: How to Remove Intractability Assumptions. In *Proceedings of the Twentieth Annual ACM Symposium on Theory of Computing*, STOC '88, pages 113–131, New York, NY, USA, 1988. ACM. doi:10.1145/62212.62223.

6 Stephen Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

7 B.S. Cirel'son. Quantum generalizations of Bell's inequality. *Lett Math Phys*, 4(93), 1980. doi:10.1007/BF00417500.

8 Frédéric Dupuis, Nicolas Gisin, Avinatan Hassidim, André Allan Méthot, and Haran Pilpel. No nonlocal box is universal. *Journal of Mathematical Physics*, 48(8):082107, 2007. doi:10.1063/1.2767538.

9 Manuel Forster and Stefan Wolf. Bipartite units of nonlocality. *Phys. Rev. A*, 84:042112, October 2011. doi:10.1103/PhysRevA.84.042112.

10 Jon Hudson. Could someone explain quantum entanglement to me like I'm 5 years old (in Reply to). *https://www.quora.com/Could-someone-explain-quantum-entanglement-to-me-like-Im-5-years-old, QUORA*, May 2018. URL: https://www.quora.com/Could-someone-explain-quantum-entanglement-to-me-like-Im-5-years-old.

11 Tsuyoshi Ito, Hirotada Kobayashi, Daniel Preda, Xiaoming Sun, and Andrew Chi-Chih Yao. Generalized Tsirelson Inequalities, Commuting-Operator Provers, and Multi-prover Interactive Proof Systems. In *Proceedings of the 23rd Annual IEEE Conference on Computational Complexity, CCC 2008, 23-26 June 2008, College Park, Maryland, USA*, pages 187–198. IEEE Computer Society, 2008. doi:10.1109/CCC.2008.12.

12 N. David Mermin. Simple unified form for the major no-hidden-variables theorems. *Phys. Rev. Lett.*, 65:3373–3376, December 1990. doi:10.1103/PhysRevLett.65.3373.

13 Asher Peres. Incompatible results of quantum measurements. *Physics Letters A*, 151(3):107–108, 1990. doi:10.1016/0375-9601(90)90172-K.

14 Sandu Popescu and Daniel Rohrlich. Quantum nonlocality as an axiom. *Foundations of Physics*, 24(3):379–385, 1994. doi:10.1007/BF02058098.

15 Stephanie Wehner. Tsirelson bounds for generalized Clauser-Horne-Shimony-Holt inequalities. *Physical Review A*, 73(2):022110, 2006.

On the Qubit Routing Problem

Alexander Cowtan
Cambridge Quantum Computing Ltd, 9a Bridge Street, Cambridge, CB2 1UB, United Kingdom

Silas Dilkes
Cambridge Quantum Computing Ltd, 9a Bridge Street, Cambridge, CB2 1UB, United Kingdom

Ross Duncan
Cambridge Quantum Computing Ltd, 9a Bridge Street, Cambridge, CB2 1UB, United Kingdom
University of Strathclyde, 26 Richmond Street, Glasgow, G1 1XH, United Kingdom
ross.duncan@cambridgequantum.com

Alexandre Krajenbrink
Cambridge Quantum Computing Ltd, 9a Bridge Street, Cambridge, CB2 1UB, United Kingdom
Laboratoire de Physique de l'École Normale Supérieure, PSL University, CNRS, Sorbonne Universités, 24 rue Lhomond, 75231 Paris Cedex 05, France

Will Simmons
Cambridge Quantum Computing Ltd, 9a Bridge Street, Cambridge, CB2 1UB, United Kingdom

Seyon Sivarajah
Cambridge Quantum Computing Ltd, 9a Bridge Street, Cambridge, CB2 1UB, United Kingdom

Abstract

We introduce a new architecture-agnostic methodology for mapping abstract quantum circuits to realistic quantum computing devices with restricted qubit connectivity, as implemented by Cambridge Quantum Computing's t|ket⟩ compiler. We present empirical results showing the effectiveness of this method in terms of reducing two-qubit gate depth and two-qubit gate count, compared to other implementations.

2012 ACM Subject Classification Theory of computation → Quantum computation theory; Computer systems organization → Quantum computing; Hardware → Quantum computation; Software and its engineering → Compilers; Software and its engineering → Retargetable compilers

Keywords and phrases Quantum Computing, Qubit routing, Compilation

Digital Object Identifier 10.4230/LIPIcs.TQC.2019.5

Related Version A full version of the paper is available at `https://arxiv.org/abs/1902.08091`.

Acknowledgements We thank Steven Herbert for many helpful conversations and encouragement.

1 Introduction

There is a significant gap between the theoretical literature on quantum algorithms and the way that quantum computers are implemented. The simple and popular *quantum circuit model* presents the quantum computer as a finite number of qubits upon which quantum gates act; see Fig. 1 for an example. Typically gates act on one or two-qubits at a time, and the circuit model allows multi-qubit gates to act on any qubits without restriction. However, in realistic hardware the qubits are typically laid out in a fixed two or three dimensional topology where gates may only be applied between neighbouring qubits. In order for a circuit to be executed on such hardware, it must be modified to ensure that whenever two-qubits are required to interact, they are adjacent in memory. This is a serious departure from the von Neumann architecture of classical computers, where operations may involve data at distant locations in memory without penalty.

We refer to the task of modifying a circuit to conform to the memory layout of a specific quantum computer as the *qubit routing problem*. When non-adjacent qubits are required to interact we can insert additional SWAP gates to exchange a qubit with a neighbour, moving it closer to its desired partner. In general many – or even all – of the qubits may need to be swapped, making this problem non-trivial. Since quantum algorithms are usually designed without reference to the connectivity constraints of any particular hardware, a solution to the routing problem is required before a quantum circuit can be executed. Therefore qubit routing forms a necessary stage of any compiler for quantum software. Current quantum computers – the so-called NISQ[1] devices – impose additional constraints. Their short coherence times and relatively low fidelity gates require that the circuit depth and the total number of gates are both as low as possible. As routing generally introduces extra gates into a circuit, increasing its size and depth, it is crucial that the circuit does not grow too much, or its performance will be compromised.

The general case of the routing problem, also called the qubit allocation problem, is known to be infeasible. The sub-problem of assigning logical qubits to physical ones is equivalent to sub-graph isomorphism [14], while determining the optimal swaps between assignments is equivalent to token-swapping [19] which is at least NP-hard [3] and possibly PSPACE-complete [10]. Siraichi et al. [14] propose an exact dynamic programming method (with complexity exponential in the number of qubits) and a heuristic method which approximates it well, at least on the small (5 qubit) circuits considered. Zulehner et al. [20] propose an algorithm based on depth partitioning and A* search which is specialised to the architectures of IBM devices [1]. Other approaches take advantage of the restricted topology typically found in quantum memories such as linear nearest neighbour [7] or hypercubic [4] which rely on classical sorting networks; see Appendix A for a discussion of this approach. Lower bound results for routing are presented by Herbert [5].

In this paper we describe the solution to the routing problem implemented in t|ket⟩, a platform-independent compiler developed by Cambridge Quantum Computing Ltd[2]. The heuristic method in t|ket⟩ matches or beats the results of other circuit mapping systems in terms of depth and total gate count of the compiled circuit, and has much reduced run time allowing larger circuits to be routed.

Aside from qubit routing, t|ket⟩ also provides translation from general circuits to any particular hardware-supported gate set, a variety of advanced circuit optimisation routines, and support for most of the major quantum software frameworks. These will be described in future papers. Compilation through t|ket⟩ guarantees hardware compatibility and minimises the depth and gate count of the final circuit across a range of hardware and software platforms.

In Section 2 we formalise the problem and present an example instance. In Section 3 we describe the method used by t|ket⟩ to solve the problem. In Section 4 we describe some of the architectures on which we tested the algorithm and in Section 5 we present empirical results of t|ket⟩'s performance, both in terms of scaling and in comparison to other compiler software. Full tables of results are provided in the Appendix.

2 The Routing Problem

We represent a quantum computer as a graph where nodes are physical qubits and edges are the allowed two-qubit interactions[3]. Since the circuit model assumes we can realise a two-qubit gate between any pair of qubits, it is equivalent to the complete graph (Fig. 2a).

[1] "Noisy intermediate-scale quantum" devices; see [13] for a survey.
[2] t|ket⟩ is available as a python module from `https://pypi.org/project/pytket/`.
[3] We don't consider architectures with multi-qubit interactions involving more than two-qubits.

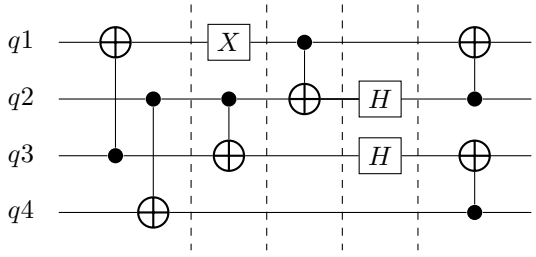


Figure 1 Example of a quantum circuit containing one and two-qubit gates acting on four qubits, $q1$, $q2$, $q3$ and $q4$. This circuit has five timesteps, each with gates acting on disjoint sets of qubits.

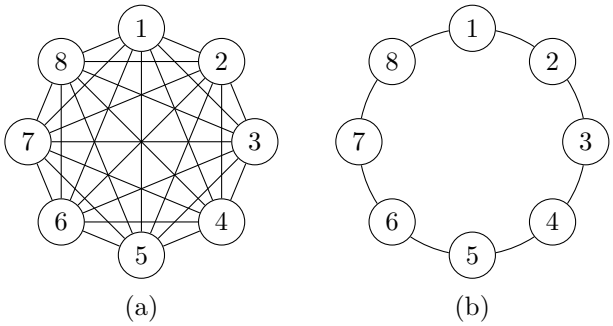


Figure 2 Nodes in the graph represent physical qubits and edges are the allowed interactions. (a) The circuit model assumes all-to-all communication between qubits, *i.e.* a complete graph and (b) a physically realistic one-dimensional nearest neighbour cyclic graph, the ring.

Realistic qubit architectures are connectivity limited: for instance, in most superconducting platforms the qubit interaction graph must be planar; ion traps present more flexibility, but are still not fully connected. For now we will use the ring graph (Fig. 2b) as a simple example. Given such a restricted graph, our goal is to emulate the complete graph with minimal additional cost.

From this point of view, the routing problem can be stated as follows. Given *(i)* an arbitrary quantum circuit and *(ii)* a connected graph specifying the allowed qubit interactions, we must produce a new quantum circuit which is equivalent to the input circuit, but uses only those interactions permitted by the specification graph. Provided the device has at least as many qubits as the input circuit then a solution always exists; our objective is to minimise the size of the output circuit.

2.1 Example: Routing on a Ring

Let's consider the problem of routing the circuit shown in Fig. 1 on the ring graph of Fig. 2(b). The first step is to divide the circuit into *timesteps*, also called *slices*. Loosely speaking, a timestep consists of a subcircuit where the gates act on disjoint sets of qubits and could in principle all be performed simultaneously (see Section 3.1 for a precise definition). The single qubit gates have no bearing on the routing problem so can be ignored, and thus a timestep can be reduced to a set of qubit pairs that are required to interact via some two-qubit gate.

Next, the logical qubits of the circuit must be mapped to the nodes of the graph. For our example a reasonable initial mapping is $q1 \to 1$, $q3 \to 2$, $q2 \to 3$, $q4 \to 4$ as shown in Fig. 3. This has the advantage that the qubits which interact in the first timestep are adjacent in the graph, and the same for the second timestep.

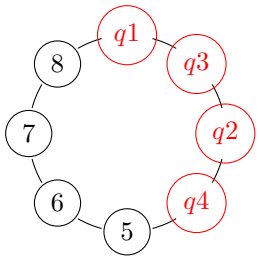


Figure 3 An initial mapping of logical qubits to nodes. Highlighted nodes are labelled with the mapped qubit.

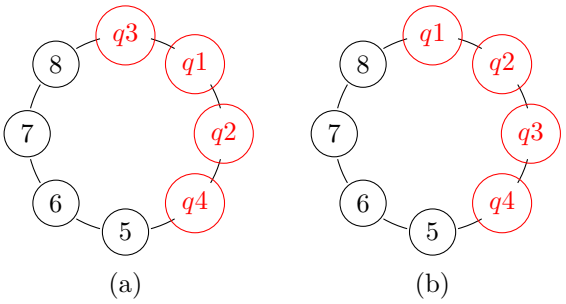


Figure 4 (a) Qubit mapping to nodes if $q1$ and $q3$ swap positions. (b) Qubit mapping to nodes if $q2$ and $q3$ swap positions.

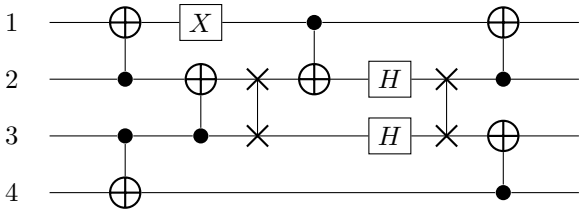


Figure 5 Quantum circuit in Fig. 1 mapped to architecture graph of Fig. 2b.

However at the third timestep our luck has run out: the CNOT gate between $q1$ and $q2$ is not possible in the current configuration. We must add SWAP gates to exchange logical qubits to enable the desired two-qubit interactions. In the example there are two candidates: swapping nodes 1 and 3, or swapping nodes 2 and 3, yielding the configurations shown in Fig. 4. Looking ahead to the final slice – slice 4 has no two-qubit gates so can be ignored – we see that $q3$ and $q4$ will need to interact. In configuration (a) these qubits are distance 3 apart, and hence two additional swaps will be needed to bring them together. In configuration (b) however they are already adjacent. As we want to minimise the number of additional elements to our circuit we choose to swap nodes 2 and 3 to yield the final circuit shown in Fig. 5.

While this was a tiny example we can see in microcosm all the key elements of the problem: the need to find a mapping of qubits to nodes; the notion of distance between qubits at the next timestep; and the need to compute the permutation of the nodes to enable the next timestep. It should be clear even from this small example that as the size of the circuit increases the number of candidate swaps increases dramatically. Further, if we have

to swap several pairs of qubits at the same time, improving the situation for one pair may worsen the situation for another pair. There is a clear arbitrage to apply to bring all the pairs together as soon as possible.

In the worst case $\mathcal{O}(n^2)$ swaps suffice to get from any n-node configuration to any other [19], although for sufficiently regular graphs much better is possible [4]. A recent lower bound result states that the minimum number of swaps is $\mathcal{O}(\log n)$ in the worst case [5], which is achieved by the cyclic butterfly network [4].

Our goal is to optimise the circuit globally so finding optimal mappings between timesteps is not sufficient. It is necessary to evaluate candidate mappings across multiple timesteps; this is the core of the $\mathsf{t}|\mathsf{ket}\rangle$ routing algorithm.

2.2 SWAP Synthesis and Routing

In the preceding section we described the routing problem in terms of inserting SWAP gates into the circuit. However not all device technologies offer SWAP as a primitive operation. Superconducting devices, for example, typically offer a single two-qubit interaction from which all other gates, including the SWAP, must be constructed. As a further complication, these interactions may be asymmetric. For example, in some IBM devices [1], the two-qubit interaction is a CNOT where one qubit is always the control and the other always the target. The graph representing the machine is therefore directed, as shown in Fig. 6, where the direction indicates the orientation of the gate.

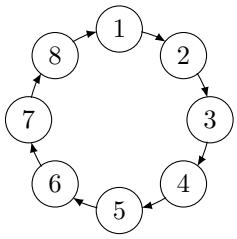


Figure 6 Architecture with one-way connectivity constraint.

This complication is easily removed by the usual trick of inserting Hadamard gates, as in Fig. 7. Hence the swap gate can be implemented by three (unidirectional) CNOTS and four Hadamards, as in Fig. 8.

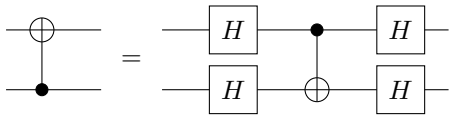


Figure 7 Inverting a CNOT gate for a directed graph.

Consider running our routed quantum circuit on the directed architecture of Fig. 6. As this graph constrains the direction of interactions, the quantum circuit we produced is no longer valid. We account for this using the inversion in Fig. 7, producing the circuit shown in Fig. 9. Many simplifications are possible on the resulting circuit, but care must be taken to ensure that the simplified circuit is still conformant to the architecture digraph.

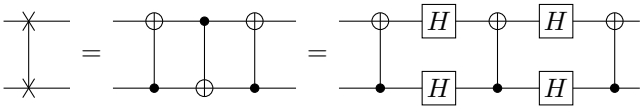


Figure 8 Representation of a SWAP gate in terms of three consecutive CNOT and its inverted representation for a directed graph.

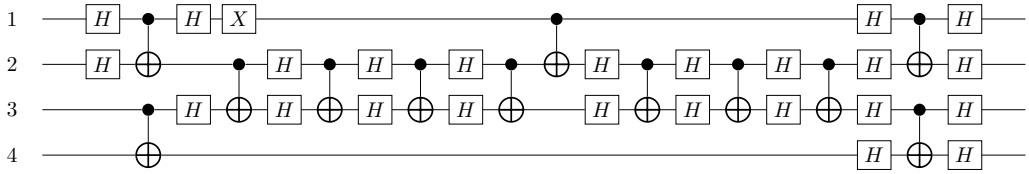


Figure 9 Quantum circuit in Fig. 1 routed for architecture graph in Fig. 6.

3 The t|ket⟩ Routing Procedure

The routing algorithm implemented in t|ket⟩ guarantees compilation of any quantum circuit to any architecture, represented as simple connected graph. It is therefore completely hardware agnostic. The algorithm proceeds in four stages: decomposing the input circuit into timesteps; determining an initial placement; routing across timesteps; and a final clean-up phase.

3.1 Slicing the Circuit Into Timesteps

Before routing we partition the quantum circuit into timesteps. The circuit structure provides a natural partial ordering of the gates; thus a greedy algorithm starting from inputs can divide the input circuit into "vertical" partitions of gates which can be executed simultaneously. We simply traverse the circuit adding the qubits involved in a two-qubit gate to the current timestep. Since only multi-qubit interactions (such as CNOT or CZ gates) constrain the problem, single-qubit gates can be ignored[4]. If a gate requires a qubit already used in the previous timestep, a new timestep is created. This procedure is repeated until all gates are assigned to a timestep. A timestep thus consists of a set of disjoint pairs of (logical) qubits which represent gates scheduled for simultaneous execution.

Applying this method to the example from Fig. 1 would yield the following timesteps.

$$\begin{aligned} 1 &\mapsto \{\ (q1, q3), (q2, q4)\ \} \\ 2 &\mapsto \{\ (q2, q3)\ \} \\ 3 &\mapsto \{\ (q1, q2), (q3, q4)\ \} \\ 4 &\mapsto \{\ (q1, q2)\ \} \end{aligned}$$

Note, that this is not the same as the illustrative slicing shown in Fig. 1!

The *density* of a timestep is a measure of the number of simultaneous gates executed. For an n-qubit architecture with single and two-qubit gates, the density is

$$d = \frac{\#\text{two-qubit gates}}{\lfloor \frac{n}{2} \rfloor} .$$

[4] More accurately: while the single-qubit gates can be ignored for the purposes of routing, they must be retained for circuit generation; for clarity we ignore them for now.

Note that $d = 1$ where every qubit is involved in a two-qubit gate in this timestep; a timestep is *sparse* when its density is close to zero. In principle, the density could be constrained to make routing easier. In practice this seems to make little difference, and we use this quantity only for the analysis in Section 5.1.

3.2 Initial Mapping

For the routing algorithm to proceed we require an initial mapping of logical qubits (referred to as qubits) and physical qubits (referred to as nodes). In $\mathsf{t}|\mathsf{ket}\rangle$ a simple but surprisingly effective procedure is used.

We iterate over the timesteps to construct a graph whose vertices are qubits. At timestep n we add the edge (q, q') to the graph if *(i)* this pair is present in the timestep and *(ii)* both qubits q and q' have degree less than 2 in the current graph. Each connected component of the resulting graph is necessarily either a line or a ring; the rings are broken by removing an arbitrarily chosen edge.

Disconnected qubits in this graph correspond either to qubits which never interact at all, or to those whose first interaction is with a qubit whose first two interactions are with others. These disconnected qubits are not included in the initial placement at all; they are added later in the routing procedure.

We then select a subgraph of the architecture with high average degree and low diameter to start from. If the architecture is Hamiltonian connected – all the common architectures are[5] – then it is possible to map the qubit graph to the architecture as one long line starting from a high degree vertex within this subgraph, and greedily choosing the highest degree available neighbour. This ensures that most of the gates in the first two timesteps can be applied without any swaps; the only exceptions are those gates corresponding to the edges removed when breaking rings.

If the initial mapping cannot be completed as one long line, then the line is split and mapped as several line segments.

3.3 Routing

The routing algorithm iteratively constructs a new circuit which conforms to the desired architecture, taking the sliced circuit and the current mapping of qubits to nodes as input.

The algorithm compares the current timestep of the input circuit to the current qubit mapping. If a gate in the current timestep requires a qubit which has not yet been mapped, it is allocated to the nearest available node to its partner. All gates which can be performed in the current mapping – all single-qubit gates and those two-qubit gates whose operands are adjacent – are immediately removed from the timestep and added to the output circuit. If this exhausts the current timestep, we advance to the next; otherwise SWAPs must be added.

We define a distance vector $d(s, m)$ which approximates the number of SWAPs needed to make timestep s executable in the mapping m; these vectors are ordered pointwise. Let s_0 denote the current timestep, s_1 for its successor, and so on, and write $\sigma \bullet m$ to indicate the action of swap σ upon the mapping m. We compute a sequence of sets of candidate SWAPs as follows:

$$\Sigma_0 = \mathsf{swaps}(s_0)$$
$$\Sigma_{t+1} = \operatorname*{arg\,min}_{\sigma \in \Sigma_t} d(s_t, \sigma \bullet m)$$

[5] See Section. 4 and Refs. [18, 8].

where $\mathsf{swaps}(s_0)$ denotes all the pertinent swaps available at the initial timestep. The sequence terminates either when $|\Sigma_t| = 1$ or after a predefined cutoff. The selected SWAP is added to the circuit and the mapping is updated accordingly. We now return to the start and continue until the entire input circuit has been consumed.

The pointwise ordering of the distance vectors employed by $\mathsf{t|ket}\rangle$ is strict in the sense that $d(s, m) > d(s, \sigma \bullet m)$ implies that for *all* pairs of qubits (q, q') in s, the longest of the shortest paths between any two paired qubits in $\sigma \bullet m$ is not longer than the longest of the shortest paths in m. In other words, the diameter of the subgraph composed of all pairs of qubits (q, q') in s should decrease strictly under the action of swap σ on the mapping m. In consequence, in some highly symmetric configurations, the algorithm sometimes gets stuck, failing to find any candidate swap. We employ two strategies to overcome this. The first is to attempt the process again with pairs of disjoint swaps instead of individual ones. If this also fails then we resort to brute force: a pair of maximally distant qubits in the current timestep are brought together using a sequence of swaps along their shortest connecting path. This guarantees at least one gate may be performed, and disrupts the symmetry of the configuration, hopefully allowing the algorithm to escape from the bad configuration.

Remark

In practice there is no need to slice the circuit in advance, and in fact better results are achieved by computing the timesteps dynamically during routing. The "next slice" is recomputed immediately after each update of the mapping, avoiding any unnecessary sequentialisation.

3.4 SWAP Synthesis and Clean-Up

If the target hardware does not support SWAP as a primitive operation, after the circuit has been routed the SWAPs in the routed circuit must be replaced with hardware appropriate gates, as per Section 2.2. While we assume that the input circuit was already well-optimised before routing, it is usually possible to remove some of the additional gates which are inserted during this process in a final clean-up pass.

The essential criterion here is that any changes to the circuit must respect the existing routing. This can be guaranteed by using any set of rewrite rules between one- and two-qubit circuits. The routing procedure will not insert SWAPs immediately before a two-qubit gate on the same two qubits, but it may do so afterwards, so the possibility to, for example, cancel consecutive CNOT gates exists. However such cancellation rules are the only "true" two-qubit rewrites which can be applied. In addition, $\mathsf{t|ket}\rangle$ uses a small set of rewrites for fusing single-qubit gates, and commuting single qubit gates past two-qubit gates. The particular rewrite rules vary according to supported gates of the hardware.

4 Graph Representation of Quantum Computers

We represent the architecture of a given quantum computer as a simple connected graph, directed or undirected. We now list some specific architecture graphs used in this work.

1. *The ring*, Fig. 2(b). A one-dimensional cyclic graph where each node is connected to its two nearest neighbors.
2. *The cyclic butterfly*, Fig. 10(a). A non-planar graph with $n = r \times 2^r$ nodes. Each node is denoted by a pair (w, i) where w is r-bit sequence corresponding to one of the 2^r rows and i represents the column. Two nodes (w, i) and (v, j) are connected if $j \equiv i + 1\,[r]$

and if $w = v$ or w and v have only one bit difference at position i, hence the connectivity is equal to 4 for any node, see Ref. [4].

3. *The square grid*, Fig. 10(b). A two-dimensional graph with a square shape where nodes are connected to their four neighbors except at the edges where there can be only two or three neighbors.

4. *The IBM Q 20 Tokyo*, Fig. 10(c). The graph supporting the 20-qubit processor produced by IBM is a two-dimensional graph with 20 nodes, it is rectangular with some extra connectivity, see Ref. [1].

5. *The Rigetti 19Q-Acorn*, Fig. 10(d). The graph supporting the quantum processor produced by Rigetti is a two-dimensional graph with 20 nodes, see Ref. [15].

In Appendix A Table 3 we present the basic properties of these graphs such as their degree and diameter, and the depth overhead of classical sorting algorithms on these graphs.

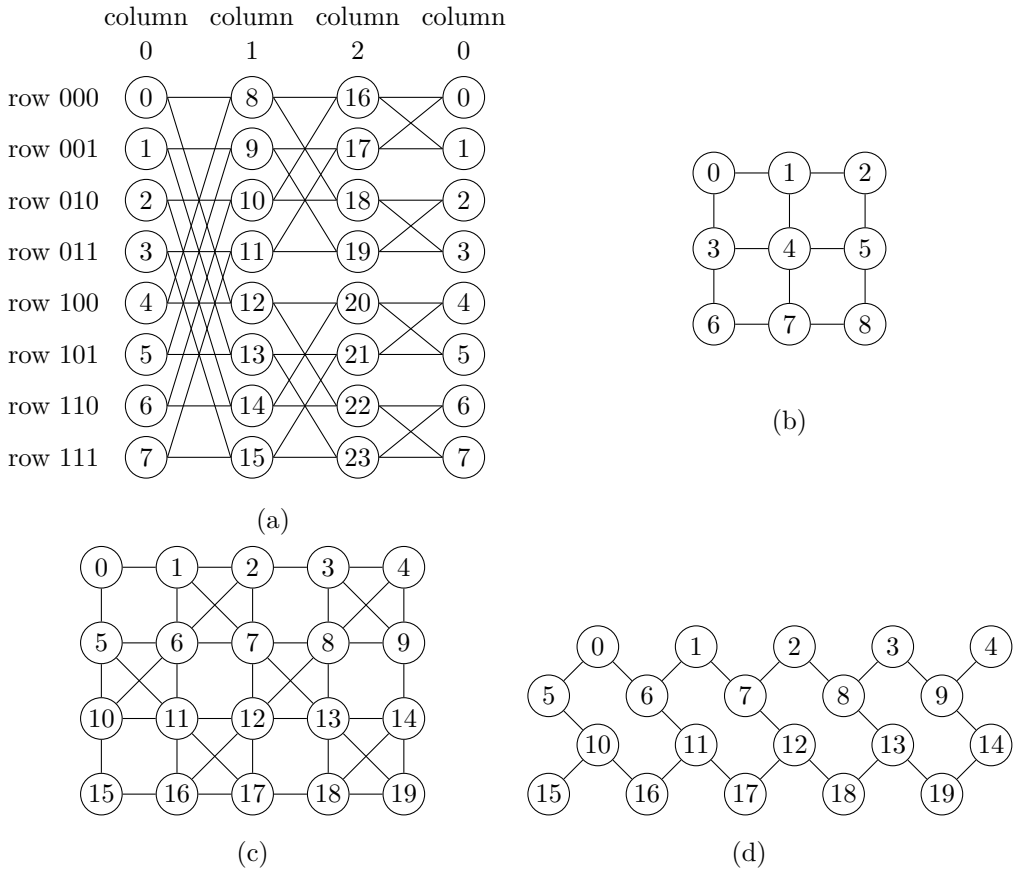


Figure 10 (a) a cyclic butterfly graph with $n = 3 \times 2^3$ nodes (the first column is represented twice to improve the readibility of the connectivity), (b) a 2-dimensional square grid with $n = 3^2$ nodes, (c) the IBM Q 20 Tokyo chip (Ref. [1]). and (d) the Rigetti 19Q-Acorn chip (Ref. [15]). The edges represent the allowed interactions between qubits.

5 Results

The current generation of quantum computers, the NISQ devices [13], are characterised by small numbers of qubits and shallow circuit depths. In this setting constant factors are more important than asymptotic analysis, so we present two sets of empirical results on the performance of t|ket⟩'s routing algorithm. In the first set of results we evaluate the scaling behaviour on synthetic inputs of increasing size. In the second we compare the performance of t|ket⟩ against competing compiler implementations on a set of realistic circuits. Note that while the t|ket⟩ algorithm is very efficient, we report on the quality of the results rather than the time or memory requirements.

5.1 Scaling

The routing algorithm described in Section 3 can handle circuits of arbitrary depth, and architectures corresponding to any connected graph. We now evaluate how increasing the circuit depth, as well as the size and connectivity of the architecture graph influence the depth of the routed circuit.

As described above, routing adds SWAP gates to the circuit, increasing both its total gate count and the depth of the circuit. Since the total gate count depends on the particular gate set supported by the architecture, we will consider only the increase in circuit depth here. Therefore a reasonable figure of merit is the depth ratio:

$$R = \frac{\text{number of output timesteps}}{\text{number of input timesteps}} ,$$

where timesteps are computed as described in Section 3.1. We define the mean depth *overhead* as

$$N = \text{number of output timesteps} - \text{number of input timesteps}.$$

For a fair comparison to classical sorting algorithms, we consider that a SWAP gate counts as only one additional gate rather than, for example, three when decomposed into CNOT gates, and hence will induce at most one additional timestep.

5.1.1 Scaling With Depth

To assess the performance of t|ket⟩ with respect to increasingly deep circuits we perform the following protocol for each of the selected architectures.

- We randomly generate 1000 circuits of density $d = 1$ and t initial timesteps for $t \in [2, 10]$. Note that requiring $d = 1$ implies there are no single-qubit gates in the circuit.
- Use t|ket⟩ to route the circuit on the chosen architecture.
- Compute R for the routed circuit.

We tested using the following five architectures:

- a ring of size $r = 64$;
- a square grid of size $r^2 = 64$;
- a cyclic butterfly of size $r2^r = 64$;
- the IBM Q 20 Tokyo ($n = 20$);
- the Rigetti 19Q-Acorn[6] ($n = 20$).

[6] The Rigetti Acorn has only 20 qubits, but due a manufacturing defect which only 19 are *usable*. This is not relevant to our tests [12].

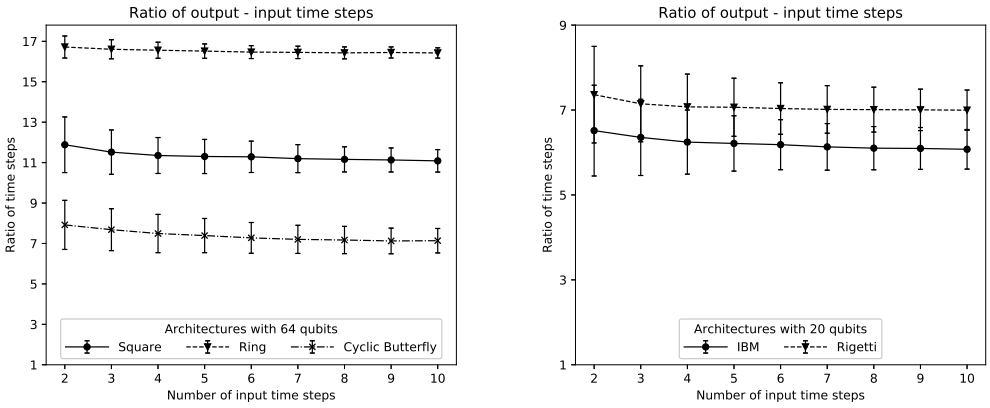


Figure 11 Multiple timesteps measurement and architecture comparison. The mean and standard deviation of the ratio R are represented. The left plot overlaps results for the ring, square grid and cyclic butterfly for 64 nodes. The right plot overlaps results for IBM and Rigetti architectures with 20 nodes. Results generated with random initial (dense) circuits with density equal to unity.

The number of nodes for the ring, square grid and cyclic butterfly architectures is chosen for fair comparison and similarly for the IBM and the Rigetti cases. To eliminate sampling bias, a single set of 64-qubit circuits was generated for the all the $n = 64$ architectures, and similarly for the $n = 20$ architectures.

Figure. 11 represents the mean and standard deviation of the ratio R for the graphs. The ratio R is approximately constant and the effect of circuit depth is dominated by the influence of the architecture's connectivity. This ratio seems to converge for circuits of depth greater than 5 and we report in Table 2 the values of R obtained for the largest number of input timesteps. While the ratios obtained seem rather large, it is worth remembering that $d = 1$ circuits are the worst case for routing.

5.1.2 Scaling With Architecture Size

To evaluate the scaling with respect to the size of the architecture we consider single-timestep random quantum circuits of varying density, which are routed on architectures of increasing size. Initial qubit mapping is disabled for these tests so that only the routing procedure is evaluated. While this is an important part of the algorithm, in this case we are interested in the scaling, to which the initial mapping only provides an initial offset.

- For each architecture of size n generate $10n$ random circuits of depth one, for each $d \in \{0.5, 0.67, 1.0\}$.
- Generate a random initial mapping of qubits on the architecture.
- Route the timestep using $\mathsf{t}|\mathsf{ket}\rangle$, using the given mapping.
- Compute N for the routed circuit.

The following architectures were evaluated:

- Rings of size $r \in [10, 200]$
- Square grids of size r^2, $r \in [3, 13]$
- Cyclic butterflies of size $r2^r$, $r \in [2, 6]$.

The results are shown in Fig. 12 and the best fit parameters are given in Table 1. The prior results for the ring and square grid are determined with a regression in log-log space

Table 1 Scaling of the depth overhead with architecture size for single-timestep random circuits.

Graph	$d = 0.5$	$d = 0.67$	$d = 1.0$
Ring	$0.2451 \times n$	$0.2451 \times n$	$0.2451 \times n$
Square	$0.5501 \times n^{0.55}$	$0.8050 \times n^{0.56}$	$0.8991 \times n^{0.58}$
Cyclic Butterfly	$0.3496 \times \log(n)^{1.85}$	$0.3002 \times \log(n)^{2.05}$	$0.1510 \times \log(n)^{2.72}$

Table 2 Summary of our scaling results for dense circuits ($d = 1$).

Graph	Depth overhead N for single-timestep circuits	Ratio output - input timesteps R
Ring	$0.2451 \times n^{1.00}$	16.42 ± 0.25 ($n = 64$)
Square grid	$0.8991 \times n^{0.58}$	11.09 ± 0.56 ($n = 64$)
Cyclic butterfly	$0.1510 \times \log(n)^{2.72}$	7.14 ± 0.61 ($n = 64$)
Rigetti 19Q-Acorn	$\emptyset$	7.00 ± 0.47
IBM Q 20 Tokyo	$\emptyset$	6.08 ± 0.47

and the cyclic butterfly in log - log(log) space (represented in the insets for $d = 1$). In each case we see that the overhead appears to grow with the diameter of the graph, although with an exponent that varies (slightly) with the density.

5.2 Realistic Benchmarks

Random circuits have an essentially uniform structure, which circuits arising from quantum algorithms typically lack. In certain cases this can make random circuits easier to route – although in the preceding section we have largely avoided this by using circuits of high density. To give $\mathsf{t}|\mathsf{ket}\rangle$ a more realistic test we have also evaluated its performance on a standard set of 156 circuits which perform various algorithms. These range in size from 6 to 16 qubits, and 7 to more than half a million gates.

We ran $\mathsf{t}|\mathsf{ket}\rangle$ on each circuit of the benchmark set, with the 16-qubit ibmqx5 *Rueschlikon*, which is a 2×8 rectangular grid, as the target architecture. We then repeated the same test set using the 20-qubit IBM Tokyo as the target architecture. Since both these architectures have CNOT as their only two-qubit operation, and since it has lower fidelity than the single-qubit operations, we selected figures of merit based on minimising the CNOT count and depth of the output circuit. In this test we do perform SWAP synthesis, to get a more realistic evaluation of the output for these devices. Let $C_{CX}(c)$ be the total number of CNOT gates in circuit c, and let $D_{CX}(c)$ be the depth of the circuit counting only the CNOT gates. The two measures of interest are

$$R_D = \frac{D_{CX}(\mathsf{out})}{D_{CX}(\mathsf{in})} \qquad R_C = \frac{C_{CX}(\mathsf{out})}{C_{CX}(\mathsf{in})}$$

where in and out are the input and output circuits respectively. The results are shown in Fig. 13. We can see that $\mathsf{t}|\mathsf{ket}\rangle$ achieves approximately linear overhead across the entire test

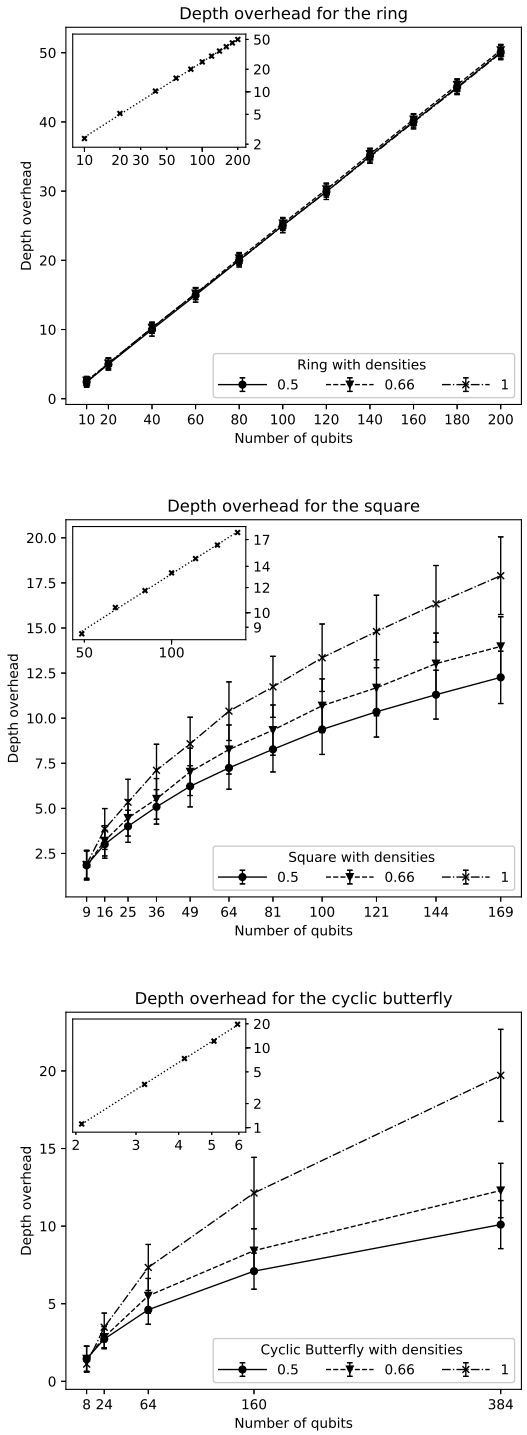


Figure 12 Variation of depth overhead with architecture size for single timestep random circuits. Plots from left to right: the ring, square and cyclic butterfly architectures. The mean and standard deviation of the depth overhead versus number of nodes (or qubits) is represented. The inset plots represent the log-log linear fit for the ring and the square (resp. log-loglog fit for the butterfly) for the data set of density $d = 1$.

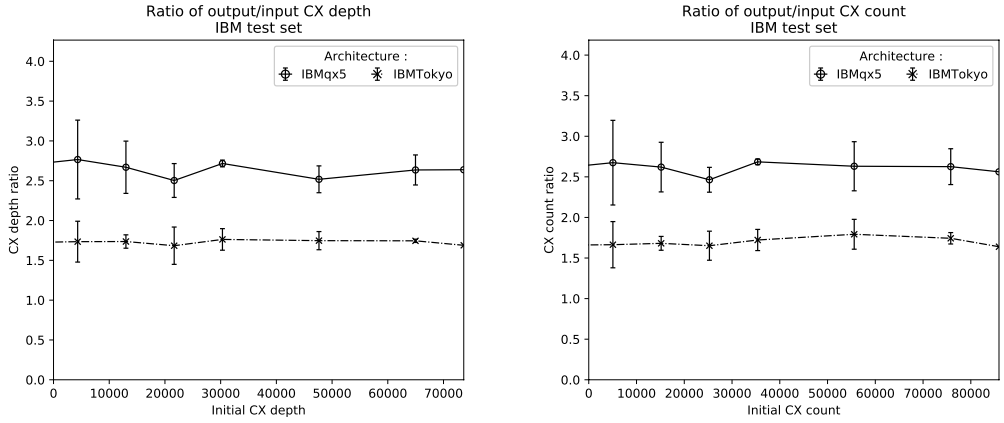


Figure 13 Performance of t|ket⟩ on realistic test examples. (left) Mean ratio of output to input CX depth as a function of circuit depth (averaged in bins) (right) Mean ratio of output to input CX count (averaged in bins).

set. The mean R_D of 2.64 and R_C of 2.61 for ibmqx5, and a mean R_D of 1.73 and R_C of 1.69 for IBM Tokyo.

We also compared the performance of t|ket⟩ to a selection of other freely available quantum compiler systems: IBM's QISKit [9], Project Q [16], and Rigetti Computing's Quilc [15][7]. None of the other compilers was able to complete the test set in the time allotted, despite being given at least an hour of compute time per example on a powerful computer[8]. For comparison, t|ket⟩ completed the entire benchmark set in 15 mins on the same hardware. In addition, Project Q does not support routing for the IBM Tokyo architecture due to its unusual graph structure; therefore it was only tested on the ibmqx5 architecture. Therefore comparison of all four compilers is only available for circuits of fewer than 2000 total gates. The comparative results are shown in Fig. 14. We can see that t|ket⟩, Qiskit and Quilc exhibit approximately linear overhead, while Project Q appears somewhat worse than linear. A line of best fit calculated using the least squares method is shown for each compiler in Fig. 14. Quilc and t|ket⟩ exhibit very similar performance; the others show significantly higher overhead.

Finally, we compared the results to the published data of Zulehner et al. [20] who use the same benchmark set, but use total gate count and depth as the metric. Since Quilc does not generate the same gate set as the others, it was excluded from this comparison. The algorithm of Zulehner et al. [20] achieves comparable performance to t|ket⟩. The results are presented in Appendix B.

The test set we used for this work was published by IBM as part of the QISKit Developer Challenge[9], a public competition to design a better routing algorithm. The competition was won by Zulehner et al. [20]. The test circuits are available from `http://iic.jku.at/eda/research/ibm_qx_mapping/`.

[7] Since Quilc emits CZ as its preferred two-qubit gate we computed its figures using D_{CZ} and C_{CZ} instead.
[8] See Appendix B for more details.
[9] `https://qx-awards.mybluemix.net/#qiskitDeveloperChallengeAward`

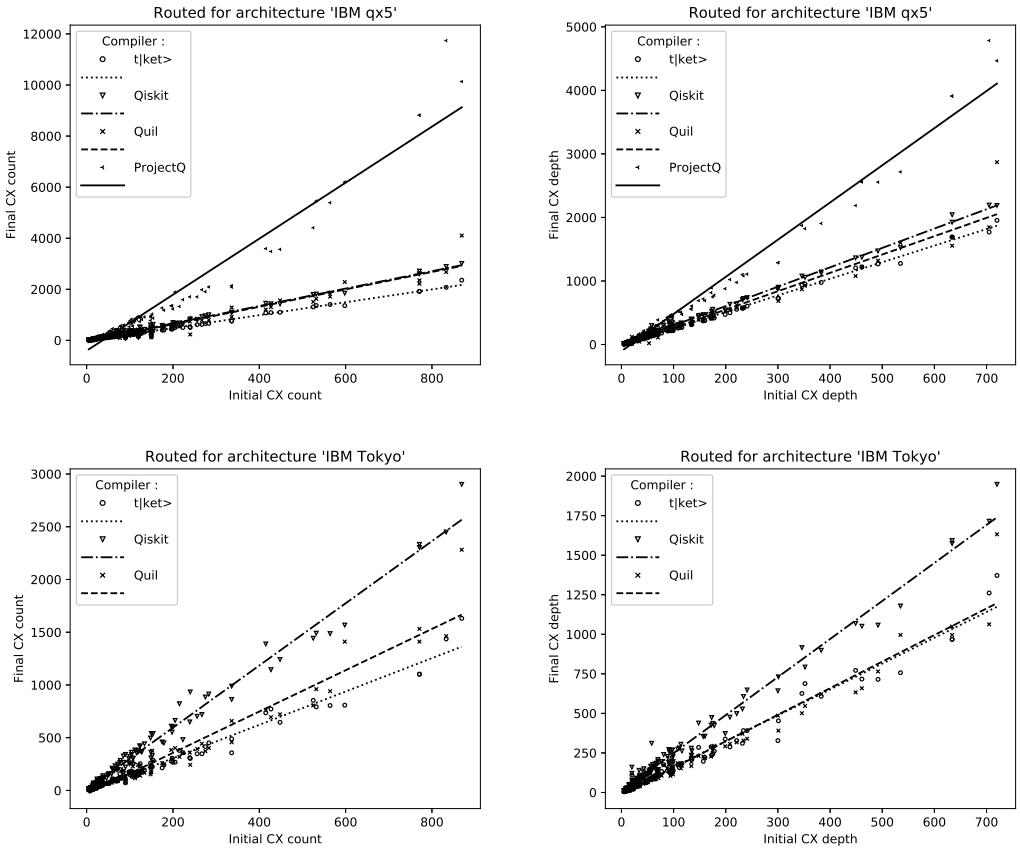


Figure 14 Comparison of performance between different compilers. Top row: routing on the ibmqx5 architecture. Bottom row: routing on the IBM Tokyo architecture. Left column: input CX count against output CX count. Right column: input CX depth against output CX depth. The benchmark is done against the test set available on `http://iic.jku.at/eda/research/ibm_qx_mapping/` and the results are averaged in bins when the initial count or depth is equal.

6 Conclusion

As better NISQ machines with the potential to effectively run quantum algorithms become available, the need for software solutions that allow users to easily run quantum circuits on them becomes more apparent. The t|ket⟩ routing module is one such solution and provides hardware compatibility with minimal extra gate overhead. It is flexible, general and scalable. In this work we have outlined how the routing procedure works and the figures of merit we use to assess routing performance for different graphs.

Finally, we consider possible extensions of this work. Firstly, we note that reinforcement learning offers an alternative approach to the qubit routing problem [6]. Eventually we foresee implementing several approaches to routing in t|ket⟩ to best adapt to differing algorithms and architectures.

Secondly, when considering the routing problem, we made the implicit assumption that all gates were equal. In real devices, notably superconducting devices, each gate has its own fidelity and run time and this has to be taken into account. Splitting a quantum circuit into timesteps becomes more complex as we introduce the different run times and we also have

to ensure that the overhead in the error rate encountered by qubits is as small as possible. Additionally, in real life experiments, it has been observed in [17] and [11] that even the properties of the qubits can fluctuate intra-days. This calls for a general protocol that could accommodate this constraint. Addressing these different constraints transforms the problem from a *routing* one to a *scheduling* one, which we plan to address with t|ket⟩. Implementing these constraints and measuring t|ket⟩ performance on this matter will be the object of future work.

References

1 IBM Q. URL: https://www.research.ibm.com/ibm-q/.

2 Robert Beals, Stephen Brierley, Oliver Gray, Aram W. Harrow, Samuel Kutin, Noah Linden, Dan Shepherd, and Mark Stather. Efficient Distributed Quantum Computing. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Science*, 469(2153), 2013. doi:10.1098/rspa.2012.0686.

3 Édouard Bonnet, Tillmann Miltzow, and Paweł Rzążewski. Complexity of Token Swapping and Its Variants. *Algorithmica*, 80(9):2656–2682, September 2018. doi:10.1007/s00453-017-0387-0.

4 Stephen Brierley. Efficient implementation of Quantum circuits with limited qubit interactions. *Quantum Information and Computation*, 17(13-14):1096–1104, 2015. arXiv:1507.04263.

5 Steven Herbert. On the depth overhead incurred when running quantum algorithms on near-term quantum computers with limited qubit connectivity. *arXiv preprint*, 2018. arXiv:1805.12570.

6 Steven Herbert and Akash Sengupta. Using Reinforcement Learning to find Efficient Qubit Routing Policies for Deployment in Near-term Quantum Computers. *arXic preprint*, 2018. arXiv:1812.11619.

7 Yuichi Hirata, Masaki Nakanishi, Shigeru Yamashita, and Yasuhiko Nakashima. An efficient conversion of quantum circuits to a linear nearest neighbor architecture. *Quantum Information and Computation*, 11:142–166, January 2011.

8 Shien-Ching Hwang and Gen-Huey Chen. Cycles in butterfly graphs. *Networks: An International Journal*, 35(2):161–171, 2000.

9 IBM Research. Qiskit. URL: https://qiskit.org.

10 Mark R. Jerrum. The complexity of finding minimum-length generator sequences. *Theoretical Computer Science*, 36:265–289, 1985. doi:10.1016/0304-3975(85)90047-7.

11 P. V. Klimov, J. Kelly, Z. Chen, M. Neeley, A. Megrant, B. Burkett, R. Barends, K. Arya, B. Chiaro, Yu Chen, A. Dunsworth, A. Fowler, B. Foxen, C. Gidney, M. Giustina, R. Graff, T. Huang, E. Jeffrey, Erik Lucero, J. Y. Mutus, O. Naaman, C. Neill, C. Quintana, P. Roushan, Daniel Sank, A. Vainsencher, J. Wenner, T. C. White, S. Boixo, R. Babbush, V. N. Smelyanskiy, H. Neven, and John M. Martinis. Fluctuations of Energy-Relaxation Times in Superconducting Qubits. *Phys. Rev. Lett.*, 121:090502, August 2018. doi:10.1103/PhysRevLett.121.090502.

12 J. S. Otterbach, R. Manenti, N. Alidoust, A. Bestwick, M. Block, B. Bloom, S. Caldwell, N. Didier, E. Schuyler Fried, S. Hong, P. Karalekas, C. B. Osborn, A. Papageorge, E. C. Peterson, G. Prawiroatmodjo, N. Rubin, Colm A. Ryan, D. Scarabelli, M. Scheer, E. A. Sete, P. Sivarajah, Robert S. Smith, A. Staley, N. Tezak, W. J. Zeng, A. Hudson, Blake R. Johnson, M. Reagor, M. P. da Silva, and C. Rigetti. Unsupervised Machine Learning on a Hybrid Quantum Computer. *arXiv.org*, 2017. arXiv:1712.05771.

13 John Preskill. Quantum Computing in the NISQ era and beyond. *Quantum*, 2:79, August 2018. doi:10.22331/q-2018-08-06-79.

14 Marcos Yukio Siraichi, Vinícius Fernandes dos Santos, Sylvain Collange, and Fernando Magno Quintão Pereira. Qubit allocation. In *Proceedings of the 2018 International Symposium on Code Generation and Optimization*, pages 113–125. ACM, 2018.

15 Robert S. Smith, Michael J. Curtis, and William Zeng. A Practical Quantum Instruction Set Architecture. Technical report, Rigetti Computing, 2016. arXiv:1608.03355.

16 Damian Steiger and Thomas Häner. Project Q: Powerful open source software for quantum computing. URL: https://projectq.ch [cited 28 January 2019].

17 Swamit S. Tannu and Moinuddin K.Qureshi. A Case for Variability-Aware Policies for NISQ-Era Quantum Computers. *arXiv.org*, 2018. arXiv:1805.10224.

18 Stephen A Wong. Hamilton cycles and paths in butterfly graphs. *Networks*, 26(3):145–150, 1995.

19 Katsuhisa Yamanaka, Erik D. Demaine, Takehiro Ito, Jun Kawahara, Masashi Kiyomi, Yoshio Okamoto, Toshiki Saitoh, Akira Suzuki, Kei Uchizawa, and Takeaki Uno. Swapping Labeled Tokens on Graphs. In Alfredo Ferro, Fabrizio Luccio, and Peter Widmayer, editors, *Fun with Algorithms*, pages 364–375. Springer International Publishing, 2014. doi:10.1007/978-3-319-07890-8_31.

20 Alwin Zulehner, Alexandru Paler, and Robert Wille. An Efficient Methodology for Mapping Quantum Circuits to the IBM QX Architectures. *arXiv.org*, 2017. arXiv:1712.04722.

21 Alwin Zulehner and Robert Wille. Compiling SU(4) Quantum Circuits to IBM QX Architectures. *arXiv.org*, 2018. arXiv:1808.05661.

A Dynamical Routing Versus Sorting Networks

The routing problem described in this work can be solved using classical sorting algorithms. One of these is the cyclic odd-even sort for the ring of Fig. 2b). Starting from an architecture with n nodes, one compares sequentially all even and odd labeled edges. After exactly $n-1$ timesteps, the input will be sorted regardless of input.

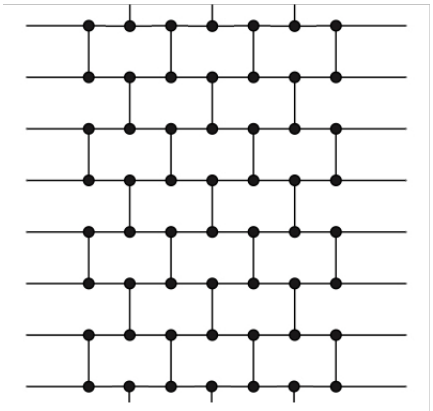


Figure 15 An example of sorting network on 8 inputs : odd-even sort over a ring.

For the ring, square and cyclic butterfly graphs presented in Section 4, we summarize in Table 3 some details on the degree and diameter these graphs and the depth overhead of classical sorting algorithms (precisely the quantity N introduced in Section 5).

The downside of classical sorting algorithms is that they are unadapative: they compute the same sequence of comparisons regardless of input. As circuits are usually sparse, see Section 3.1, this leaves many unecessary comparisons, and would treat quantum circuits as sequences of hard timesteps. Indeed, routing solutions derived from classical sorting algorithms tend to pack a quantum circuit into multiple timesteps and then insert SWAP gates in between timesteps. Solving the routing problem sequentially timestep by timestep produces a concatenation of locally optimal solutions which can be very far from the globally optimal one. A good solution should be dynamic, consider a SWAP gate's influence on multiple timesteps, and optimize the global problem rather than the local one. See Ref. [21] for an additional discussion on this matter. Additional details on sorting networks in quantum computing are available in Ref. [2, 4].

Table 3 Comparison of different networks with n nodes.

Graph	Degree	Diameter	N
Ring	2	$\frac{n-1}{2}$	$n-1$
Square grid	4	$2\sqrt{n}-1$	$3\sqrt{n}$
Cyclic butterfly ($n = r \times 2^r$)	4	$\frac{3\log_2(n)}{2}$	$6\log_2(n)$

B Detailed Benchmark Results

The table rows are the names of the benchmark QASM circuits, which are available from `www.github.com/iic-jku/ibm_qx_mapping`. Benchmark data for Zulehner et al. is collected from results presented in their paper [20] – note they do not present data for the complete set of examples. An example Jupyter workbook which demonstrates the benchmarking procedure is found at `https://github.com/CQCL/pytket/blob/master/examples/tket_benchmarking.ipynb`.

All computations were run on a Google Cloud virtual machine with the following specification: machine type n1-standard-2 (2 vCPUs, 7.5GB Memory), Intel Broadwell, 16GB RAM and Standard Persistent Disk. Each example was run until completion, the computation aborted, or until 60 minutes of real time had passed, whichever came first. Note that Quilc aborts in much less than 60 minutes.

In the tables, g indicates the gate count of the circuit; in Table 4 this means all gates; in Table 5 and 6 this means CX count only. The circuit depth is labelled d; in Table 4 this means total depth; in Table 5 and 6 this means CX depth only. The bold values are the best performance on the each row. The "t|ket⟩ comparison" column shows the ratio between t|ket⟩'s performance and the best other compiler; values less than 1 indicate that t|ket⟩ performs better.

NOTE

The example circuit "ground_state_estimation" gives anomalously low values after routing. This is due to an error in the circuit, which allows the post-routing clean-up pass of t|ket⟩ to eliminate almost the entire circuit.

B.1 All Gates Comparison on ibmqx5

Table 4 All gates comparison on ibmqx5.

			Qiskit 0.7.0		Zulehner et al.		CQC's t|ket⟩		t|ket⟩ comparison	
Name	g_{in}	d_{in}	g_{out}	d_{out}	g_{out}	d_{out}	g_{out}	d_{out}	r_{gate}	r_{depth}
xor5_254	7	5	45	24	*	*	**25**	**14**	0.56	0.58
graycode6_47	5	5	**13**	**7**	*	*	15	9	1.15	1.29
ex1_226	7	5	56	32	*	*	**25**	**14**	0.45	0.44
4gt11_84	18	11	59	34	*	*	**48**	**32**	0.81	0.94
4mod5-v0_20	20	12	64	35	*	*	**50**	**31**	0.78	0.89
ex-1_166	19	12	**53**	36	*	*	**53**	**34**	1.00	0.94
4mod5-v1_22	21	12	75	46	*	*	**64**	**40**	0.85	0.87
mod5d1_63	22	13	94	53	*	*	**65**	**39**	0.69	0.74
ham3_102	20	13	62	39	*	*	**47**	**32**	0.76	0.82
4gt11_83	23	16	100	57	*	*	**75**	**46**	0.75	0.81
4gt11_82	27	20	109	61	*	*	**86**	**55**	0.79	0.90
rd32-v0_66	34	20	116	73	*	*	**70**	**46**	0.60	0.63
alu-v0_27	36	21	163	86	*	*	**96**	**61**	0.59	0.71
4mod5-v1_24	36	21	142	80	*	*	**97**	**62**	0.68	0.78
4mod5-v0_19	35	21	143	88	*	*	**103**	**66**	0.72	0.75
mod5mils_65	35	21	137	84	*	*	**93**	**59**	0.68	0.70

			Qiskit 0.7.0		Zulehner et al.		CQC's t|ket⟩		t|ket⟩ comparison	
Name	g_{in}	d_{in}	g_{out}	d_{out}	g_{out}	d_{out}	g_{out}	d_{out}	r_{gate}	r_{depth}
rd32-v1_68	36	21	130	76	*	*	**70**	**46**	0.54	0.61
alu-v1_28	37	22	142	84	*	*	**99**	**66**	0.70	0.79
alu-v2_33	37	22	138	80	*	*	**91**	**57**	0.66	0.71
alu-v4_37	37	22	130	75	*	*	**103**	**64**	0.79	0.85
alu-v3_35	37	22	143	79	*	*	**103**	**64**	0.72	0.81
3_17_13	36	22	127	88	*	*	**89**	**62**	0.70	0.70
alu-v1_29	37	22	135	78	*	*	**101**	**66**	0.75	0.85
miller_11	50	29	158	107	*	*	**139**	**92**	0.88	0.86
alu-v3_34	52	30	212	126	*	*	**146**	**99**	0.69	0.79
decod24-v2_43	52	30	206	123	*	*	**136**	**89**	0.66	0.72
decod24-v0_38	51	30	173	108	*	*	**127**	**84**	0.73	0.78
mod5d2_64	53	32	185	110	*	*	**158**	**105**	0.85	0.95
4gt13_92	66	38	263	158	*	*	**187**	**119**	0.71	0.75
4gt13-v1_93	68	39	281	164	*	*	**168**	**105**	0.60	0.64
4mod5-v0_18	69	40	272	160	*	*	**181**	**122**	0.67	0.76
decod24-bdd_294	73	40	262	147	*	*	**205**	**129**	0.78	0.88
one-two-three-v2_100	69	40	256	151	*	*	**194**	**129**	0.76	0.85
one-two-three-v3_101	70	40	268	165	*	*	**199**	**129**	0.74	0.78
4mod5-v1_23	69	41	283	154	*	*	**196**	**136**	0.69	0.88
4mod5-bdd_287	70	41	283	152	*	*	**212**	**133**	0.75	0.88
rd32_270	84	47	289	167	*	*	**220**	**148**	0.76	0.89
4gt5_75	83	47	315	175	*	*	**227**	**143**	0.72	0.82
alu-bdd_288	84	48	319	170	*	*	**253**	**160**	0.79	0.94
alu-v0_26	84	49	331	188	*	*	**230**	**145**	0.69	0.77
decod24-v1_41	85	50	355	198	*	*	**221**	**139**	0.62	0.70
rd53_138	132	56	642	269	*	*	**416**	**217**	0.65	0.81
4gt5_76	91	56	360	195	*	*	**266**	**168**	0.74	0.86
4gt13_91	103	61	410	229	*	*	**269**	**181**	0.66	0.79
cnt3-5_179	175	61	684	275	*	*	**569**	**221**	0.83	0.80
qft_10	200	63	692	214	447	170	**345**	**154**	0.77	0.91
4gt13_90	107	65	427	250	*	*	**284**	**190**	0.67	0.76
alu-v4_36	115	66	410	232	*	*	**286**	**184**	0.70	0.79
mini_alu_305	173	69	829	317	**474**	**225**	524	243	1.11	1.08
ising_model_10	480	70	**235**	**41**	251	47	255	**41**	1.09	1.00
ising_model_16	786	71	**391**	**41**	426	48	426	**41**	1.09	1.00
ising_model_13	633	71	**313**	**41**	329	47	398	110	1.27	2.68
4gt5_77	131	74	451	266	*	*	**356**	**226**	0.79	0.85
sys6-v0_111	215	75	975	365	**613**	**250**	675	264	1.10	1.06
one-two-three-v1_99	132	76	501	299	*	*	**354**	**233**	0.71	0.78
one-two-three-v0_98	146	82	578	343	*	*	**376**	**250**	0.65	0.73
decod24-v3_45	150	84	551	318	*	*	**383**	**247**	0.70	0.78
4gt10-v1_81	148	84	555	310	*	*	**377**	**248**	0.68	0.80

			Qiskit 0.7.0		Zulehner et al.		CQC's t|ket⟩		t|ket⟩ comparison	
Name	g_{in}	d_{in}	g_{out}	d_{out}	g_{out}	d_{out}	g_{out}	d_{out}	r_{gate}	r_{depth}
aj-e11_165	151	86	541	309	*	*	**402**	**260**	0.74	0.84
4mod7-v0_94	162	92	641	374	*	*	**442**	**291**	0.69	0.78
alu-v2_32	163	92	587	340	*	*	**458**	**292**	0.78	0.86
rd73_140	230	92	1008	416	**656**	**301**	675	329	1.03	1.09
4mod7-v1_96	164	94	617	351	*	*	**443**	**284**	0.72	0.81
4gt4-v0_80	179	101	704	373	*	*	**458**	**293**	0.65	0.79
mod10_176	178	101	685	384	*	*	**459**	**293**	0.67	0.76
0410184_169	211	104	846	399	758	**336**	**739**	347	0.97	1.03
qft_16	512	105	2131	532	1341	**404**	**1233**	446	0.92	1.10
4gt12-v0_88	194	108	793	432	*	*	**499**	**320**	0.63	0.74
rd84_142	343	110	1660	548	**971**	**353**	1166	506	1.20	1.43
rd53_311	275	124	1358	560	**942**	469	959	**452**	1.02	0.96
4_49_16	217	125	783	459	*	*	**610**	**398**	0.78	0.87
sym9_146	328	127	1584	640	**955**	**425**	999	451	1.05	1.06
4gt12-v1_89	228	130	855	473	*	*	**592**	**381**	0.69	0.81
4gt12-v0_87	247	131	919	484	*	*	**650**	**396**	0.71	0.82
4gt4-v0_79	231	132	893	495	*	*	**622**	**399**	0.70	0.81
hwb4_49	233	134	929	540	*	*	**621**	**405**	0.67	0.75
sym6_316	270	135	1381	625	**852**	**456**	890	471	1.04	1.03
4gt12-v0_86	251	135	992	512	*	*	**668**	**410**	0.67	0.80
4gt4-v0_72	258	137	903	485	*	*	**658**	**401**	0.73	0.83
4gt4-v0_78	235	137	935	492	*	*	**641**	**411**	0.69	0.84
mod10_171	244	139	859	496	*	*	**640**	**417**	0.75	0.84
4gt4-v1_74	273	154	1008	570	*	*	**732**	**469**	0.73	0.82
rd53_135	296	159	1132	604	*	*	**854**	**536**	0.75	0.89
mini-alu_167	288	162	953	557	*	*	**748**	**480**	0.78	0.86
one-two-three-v0_97	290	163	1060	610	*	*	**737**	**487**	0.70	0.80
ham7_104	320	185	1327	697	*	*	**896**	**550**	0.68	0.79
decod24-enable_126	338	190	1280	730	*	*	**894**	**586**	0.70	0.80
mod8-10_178	342	193	1341	722	*	*	**879**	**561**	0.66	0.78
cnt3-5_180	485	209	1833	822	**1376**	**669**	1560	819	1.13	1.22
ex3_229	403	226	1566	859	*	*	**1057**	**663**	0.67	0.77
4gt4-v0_73	395	227	1413	799	*	*	**1037**	**671**	0.73	0.84
mod8-10_177	440	251	1494	884	*	*	**1169**	**750**	0.78	0.85
C17_204	467	253	1789	950	*	*	**1318**	**812**	0.74	0.85
alu-v2_31	451	255	1499	874	*	*	**1171**	**762**	0.78	0.87
rd53_131	469	261	1875	1016	*	*	**1238**	**776**	0.66	0.76
alu-v2_30	504	285	1746	982	*	*	**1335**	**850**	0.76	0.87
mod5adder_127	555	302	2094	1135	*	*	**1407**	**903**	0.67	0.80
rd53_133	580	327	2149	1172	*	*	**1623**	**986**	0.76	0.84
cm82a_208	650	337	2624	1303	*	*	**1777**	**1036**	0.68	0.80
majority_239	612	344	2437	1267	*	*	**1648**	**1022**	0.68	0.81
ex2_227	631	355	2568	1340	*	*	**1722**	**1092**	0.67	0.81
sf_276	778	435	2895	1615	*	*	**2012**	**1312**	0.69	0.81

			Qiskit 0.7.0		Zulehner et al.		CQC's t|ket⟩		t|ket⟩ comparison	
Name	g_{in}	d_{in}	g_{out}	d_{out}	g_{out}	d_{out}	g_{out}	d_{out}	r_{gate}	r_{depth}
sf_274	781	436	2875	1577	*	*	**1989**	**1293**	0.69	0.82
con1_216	954	508	3836	1966	*	*	**2815**	**1689**	0.73	0.86
wim_266	986	514	3777	1921	2985	1711	**2861**	**1707**	0.96	1.00
rd53_130	1043	569	3783	2049	*	*	**2872**	**1756**	0.76	0.86
f2_232	1206	668	4737	2459	*	*	**3529**	**2208**	0.74	0.90
cm152a_212	1221	684	4791	2499	3738	**2155**	**3580**	2195	0.96	1.02
rd53_251	1291	712	4868	2668	*	*	**3665**	**2279**	0.75	0.85
hwb5_53	1336	758	4917	2766	*	*	**3577**	**2287**	0.73	0.83
cm42a_207	1776	940	6862	3507	5431	**3013**	**5066**	3043	0.93	1.01
pm1_249	1776	940	7274	3744	5431	**3013**	**5066**	3043	0.93	1.01
dc1_220	1914	1038	7632	3973	5946	3378	**5433**	**3180**	0.91	0.94
squar5_261	1993	1049	8019	4006	6267	**3448**	**6110**	3532	0.97	1.02
z4_268	3073	1644	12567	6352	9717	**5335**	**9379**	5532	0.97	1.04
sqrt8_260	3009	1659	12852	6615	9744	5501	**8869**	**5355**	0.91	0.97
radd_250	3213	1781	13098	6880	10441	**5872**	**9831**	5892	0.94	1.00
adr4_197	3439	1839	13966	7050	11301	6205	**10074**	**5910**	0.89	0.95
sym6_145	3888	2187	14078	7794	*	*	**10961**	**6858**	0.78	0.88
misex1_241	4813	2676	–	–	15185	**8729**	**14411**	8834	0.95	1.01
rd73_252	5321	2867	–	–	*	*	**15666**	**9288**		
cycle10_2_110	6050	3386	–	–	19857	**11141**	**18361**	11314	0.92	1.02
hwb6_56	6723	3736	–	–	*	*	**18688**	**11650**		
square_root_7	7630	3847	–	–	25212	**13205**	**23258**	13305	0.92	1.01
ham15_107	8763	4819	–	–	28310	**15891**	**26551**	15951	0.94	1.00
dc2_222	9462	5242	–	–	30680	**17269**	**29784**	18303	0.97	1.06
sqn_258	10223	5458	–	–	32095	17801	**29740**	**17456**	0.93	0.98
inc_237	10619	5863	–	–	34375	**19176**	**32096**	19523	0.93	1.02
cm85a_209	11414	6374	–	–	37746	21189	**34467**	**21014**	0.91	0.99
rd84_253	13658	7261	–	–	45497	24473	**41770**	**24147**	0.92	0.99
co14_215	17936	8570	–	–	63826	**30366**	**57906**	30807	0.91	1.01
root_255	17159	8835	–	–	57874	**30068**	**52900**	30448	0.91	1.01
mlp4_245	18852	10328	–	–	*	*	**59020**	**35251**		
urf2_277	20112	11390	–	–	*	*	**64763**	**37903**		
sym9_148	21504	12087	–	–	66637	38849	**61342**	**37448**	0.92	0.96
life_238	22445	12511	–	–	74632	41767	**67852**	**40987**	0.91	0.98
hwb7_59	24379	13437	–	–	*	*	**68463**	**41787**		
max46_240	27126	14257	–	–	84914	**46270**	**80329**	46590	0.95	1.01
clip_206	33827	17879	–	–	114336	**60882**	**104857**	61053	0.92	1.00
9symml_195	34881	19235	–	–	116508	64279	**106669**	**63651**	0.92	0.99
sym9_193	34881	19235	–	–	116508	64279	**106669**	**63651**	0.92	0.99
sao2_257	38577	19563	–	–	131002	**66975**	**120587**	68114	0.92	1.02
dist_223	38046	19694	–	–	125867	**66318**	**117367**	67731	0.93	1.02
urf5_280	49829	27822	–	–	*	*	**155513**	**92045**		
urf1_278	54766	30955	–	–	*	*	**178380**	**104583**		
sym10_262	64283	35572	–	–	215569	118753	**197690**	**118233**	0.92	1.00
hwb8_113	69380	38717	–	–	*	*	**201013**	**122660**		
urf2_152	80480	44100	–	–	*	*	**221274**	**139339**		

			Qiskit 0.7.0		Zulehner et al.		CQC's t|ket⟩		t|ket⟩ comparison	
Name	g_{in}	d_{in}	g_{out}	d_{out}	g_{out}	d_{out}	g_{out}	d_{out}	r_{gate}	r_{depth}
urf3_279	125362	70702	–	–	440509	239702	**406738**	**237181**	0.92	0.99
plus63mod4096_163	128744	72246	–	–	439981	243861	**402395**	**243814**	0.91	1.00
urf5_158	164416	89145	–	–	*	*	**465129**	**284825**		
urf6_160	171840	93645	–	–	580295	**313011**	**541013**	313653	0.93	1.00
urf1_149	184864	99585	–	–	*	*	**525310**	**321185**		
plus63mod8192_164	187112	105142	–	–	640204	**354076**	**585116**	355168	0.91	1.00
hwb9_119	207775	116199	–	–	655220	375105	**608175**	**369246**	0.93	0.98
urf3_155	423488	229365	–	–	*	*	**1211536**	**735676**		
ground_state _esti-mation_10	390180	245614	–	–	520010	376695	**12243**	**6804**	0.02	0.02
urf4_187	512064	264330	–	–	1650845	878249	**1487289**	**867091**	0.90	0.99

g: the number of quantum gates (elementary operations), d: depth of the quantum circuits, – are time-outs and * are data not provided by the Zulehner et al.

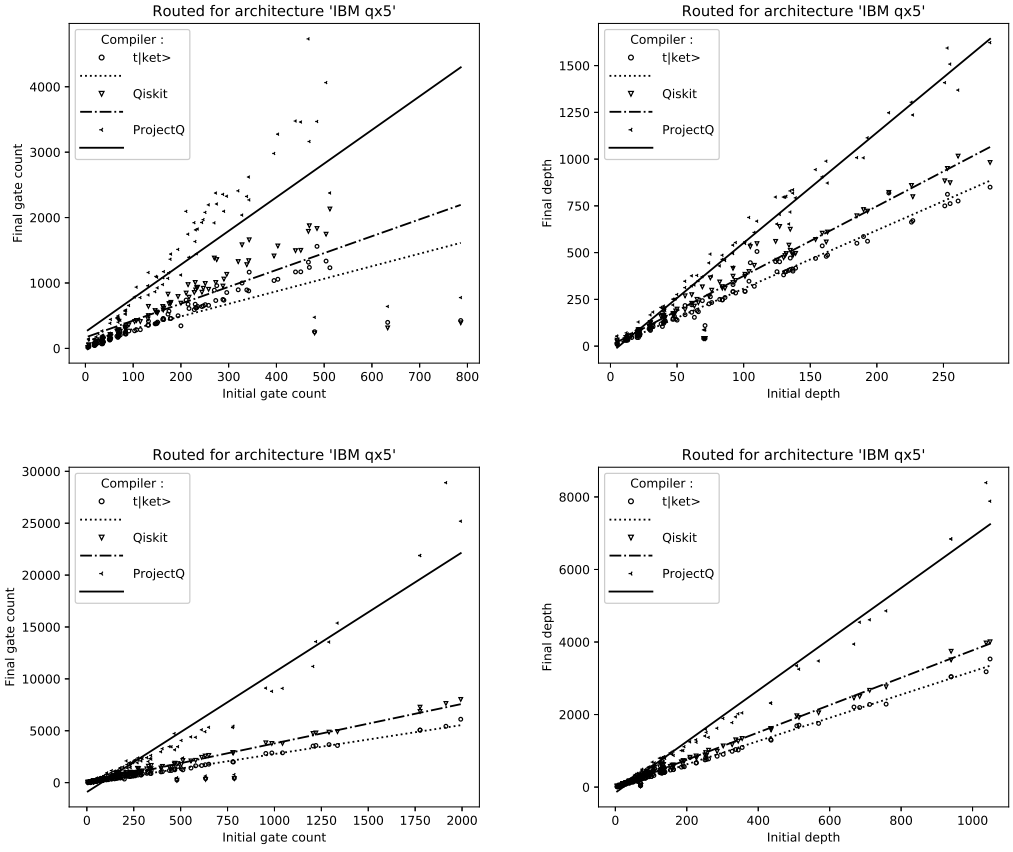


Figure 16 Routing comparison on ibmqx5, gate count and depth of the routed circuits when counting all gates. The upper charts are a zoomed in version of the initial segment of the lower charts. The results are averaged in bins when the initial count or depth is equal.

B.2 CX Only Comparison on ibmqx5

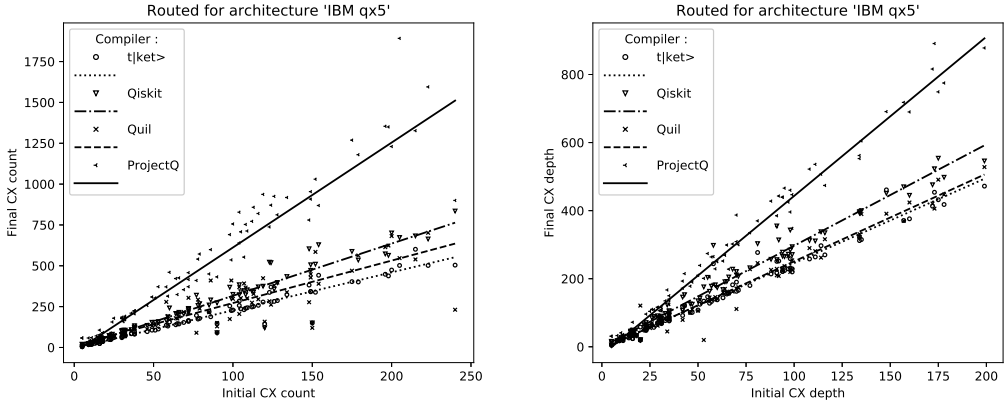


Figure 17 Routing comparison on ibmqx5, CX count and CX depth when counting only CX gates. The charts are a zoomed in version of the initial segment of upper charts of Fig. 14.

Table 5 CX gates only comparison on ibmqx5.

			Qiskit 0.7.0		Project Q 0.4.1		Quilc 1.1.1 Pyquil 2.1.1		CQC's t|ket⟩		t|ket⟩ comparison	
Name	g_{in}	d_{in}	g_{out}	d_{out}	g_{out}	d_{out}	g_{out}	d_{out}	g_{out}	d_{out}	r_{gate}	r_{depth}
xor5_254	5	5	17	14	58	31	17	12	**8**	**8**	0.47	0.67
graycode6_47	5	5	**5**	**5**	**5**	**5**	**5**	**5**	**5**	**5**	1.00	1.00
ex1_226	5	5	20	16	58	31	17	12	**8**	**8**	0.47	0.67
4gt11_84	9	8	22	18	58	33	25	18	**18**	**17**	0.82	0.94
4mod5-v0_20	10	9	22	19	43	28	20	**17**	**19**	18	0.95	1.06
ex-1_166	9	9	19	19	**18**	**18**	27	21	**18**	**18**	1.00	1.00
4mod5-v1_22	11	10	29	27	60	41	29	**21**	**23**	22	0.79	1.05
mod5d1_63	13	11	37	29	48	36	40	28	**25**	**22**	0.68	0.79
ham3_102	11	11	24	22	20	20	24	20	**18**	**18**	0.90	0.90
4gt11_83	14	14	36	30	71	38	34	28	**29**	**26**	0.85	0.93
4gt11_82	18	18	42	**33**	91	50	43	34	**36**	**33**	0.86	1.00
rd32-v0_66	16	16	43	40	36	31	39	32	**24**	**24**	0.67	0.77
alu-v0_27	17	15	60	46	71	50	52	39	**38**	**36**	0.73	0.92
4mod5-v1_24	16	15	53	44	78	51	46	38	**34**	**33**	0.74	0.87
4mod5-v0_19	16	16	53	49	106	72	44	37	**37**	**36**	0.84	0.97
mod5mils_65	16	16	52	47	43	41	46	36	**34**	**33**	0.79	0.92
rd32-v1_68	16	16	47	40	36	31	36	29	**24**	**24**	0.67	0.83
alu-v1_28	18	16	52	44	42	**36**	50	40	**39**	37	0.93	1.03
alu-v2_33	17	15	53	45	72	51	52	39	**35**	**33**	0.67	0.85
alu-v4_37	18	16	51	43	49	43	53	40	**39**	**37**	0.80	0.93
alu-v3_35	18	16	55	45	49	43	54	40	**39**	**37**	0.80	0.93
3_17_13	17	17	47	47	41	41	50	39	**35**	**35**	0.85	0.90
alu-v1_29	17	15	51	42	71	50	51	38	**38**	**36**	0.75	0.95
miller_11	23	23	57	57	52	52	77	59	**50**	**50**	0.96	0.96
alu-v3_34	24	23	81	71	156	96	73	62	**57**	**56**	0.78	0.90

			Qiskit 0.7.0		Project Q 0.4.1		Quilc 1.1.1 Pyquil 2.1.1		CQC's t|ket⟩		t|ket⟩ comparison	
Name	g_{in}	d_{in}	g_{out}	d_{out}	g_{out}	d_{out}	g_{out}	d_{out}	g_{out}	d_{out}	r_{gate}	r_{depth}
decod24-v2_43	22	22	73	65	81	67	63	53	**49**	**49**	0.78	0.92
decod24-v0_38	23	23	62	56	88	65	66	52	**48**	**48**	0.77	0.92
mod5d2_64	25	25	70	61	179	104	63	**55**	**61**	60	0.97	1.09
4gt13_92	30	26	98	85	177	110	86	69	**66**	**64**	0.77	0.93
4gt13-v1_93	30	27	104	89	185	109	79	61	**60**	**57**	0.76	0.93
4mod5-v0_18	31	31	103	90	158	109	91	70	**70**	**68**	0.77	0.97
decod24-bdd_294	32	31	97	79	170	105	98	77	**77**	**71**	0.79	0.92
one-two-three-v2_100	32	29	96	83	191	116	**69**	**62**	71	71	1.03	1.15
one-two-three-v3_101	32	29	102	91	182	116	78	**68**	**72**	69	0.92	1.01
4mod5-v1_23	32	30	106	86	178	106	102	79	**74**	**73**	0.73	0.92
4mod5-bdd_287	31	31	106	83	170	115	100	76	**82**	**75**	0.82	0.99
rd32_270	36	35	109	92	210	124	101	**82**	**84**	**82**	0.83	1.00
4gt5_75	38	33	121	98	223	135	116	83	**83**	**77**	0.72	0.93
alu-bdd_288	38	35	122	96	206	140	124	90	**95**	**88**	0.78	0.98
alu-v0_26	38	35	125	103	224	135	115	80	**83**	**78**	0.72	0.97
decod24-v1_41	38	35	133	106	225	150	118	93	**83**	**76**	0.70	0.82
rd53_138	60	42	242	146	460	193	170	**104**	**156**	118	0.92	1.13
4gt5_76	46	42	135	107	257	172	112	**79**	**97**	92	0.87	1.16
4gt13_91	49	46	154	126	309	178	131	**100**	**106**	101	0.81	1.01
cnt3-5_179	85	43	260	153	450	215	**216**	**121**	219	**121**	1.01	1.00
qft_10	90	34	273	123	432	167	**129**	**45**	147	85	1.14	1.89
4gt13_90	53	50	158	135	372	210	144	**106**	**113**	108	0.78	1.02
alu-v4_36	51	47	154	125	303	190	136	110	**106**	**101**	0.78	0.92
mini_alu_305	77	53	320	176	410	201	–	–	**196**	**134**	0.61	0.76
ising_model_10	90	20	**90**	**20**	**90**	**20**	**90**	**20**	**90**	**20**	1.00	1.00
ising_model_16	150	20	**150**	**20**	**150**	**20**	**150**	**20**	**150**	**20**	1.00	1.00
ising_model_13	120	20	**120**	**20**	**120**	**20**	**120**	**20**	144	58	1.20	2.90
4gt5_77	58	51	165	143	371	239	158	**121**	**136**	125	0.86	1.03
sys6-v0_111	98	55	369	204	691	274	279	**137**	**254**	147	0.91	1.07
one-two-three-v1_99	59	56	187	163	314	205	174	135	**131**	**128**	0.75	0.95
one-two-three-v0_98	65	59	207	182	324	229	192	145	**146**	**140**	0.76	0.97
decod24-v3_45	64	57	208	178	422	264	177	146	**139**	**133**	0.79	0.91
4gt10-v1_81	66	60	215	174	426	250	179	**138**	**144**	**138**	0.80	1.00
aj-e11_165	69	63	205	168	354	252	190	**141**	**153**	144	0.81	1.02
4mod7-v0_94	72	66	239	205	372	228	217	160	**162**	**157**	0.75	0.98
alu-v2_32	72	64	225	191	416	269	211	**157**	**165**	**157**	0.78	1.00
rd73_140	104	68	384	228	640	302	304	**170**	**257**	182	0.85	1.07
4mod7-v1_96	72	65	234	192	458	299	205	162	**160**	**155**	0.78	0.96
4gt4-v0_80	79	71	268	207	572	300	218	**162**	**178**	166	0.82	1.02
mod10_176	78	70	257	211	541	309	238	175	**174**	**163**	0.73	0.93
0410184_169	104	70	323	221	828	387	293	**171**	**284**	190	0.97	1.11
qft_16	240	58	835	298	900	265	**369**	**111**	504	244	1.37	2.20

			Qiskit 0.7.0		Project Q 0.4.1		Quilc 1.1.1 Pyquil 2.1.1		CQC's t|ket⟩		t|ket⟩ comparison	
Name	g_{in}	d_{in}	g_{out}	d_{out}	g_{out}	d_{out}	g_{out}	d_{out}	g_{out}	d_{out}	r_{gate}	r_{depth}
4gt12-v0_88	86	77	291	232	599	335	231	182	**188**	**180**	0.81	0.99
rd84_142	154	81	629	303	869	370	**410**	**228**	442	277	1.08	1.21
rd53_311	124	92	519	314	871	443	390	**246**	**370**	252	0.95	1.02
4_49_16	99	91	300	256	536	353	262	**208**	**226**	217	0.86	1.04
sym9_146	148	91	604	355	780	385	410	**229**	**382**	251	0.93	1.10
4gt12-v1_89	100	88	325	257	756	428	277	**210**	**229**	214	0.83	1.02
4gt12-v0_87	112	94	345	266	772	440	298	226	**248**	**221**	0.83	0.98
4gt4-v0_79	105	94	341	278	708	439	274	223	**236**	**222**	0.86	1.00
hwb4_49	107	99	348	294	553	348	302	229	**231**	**220**	0.76	0.96
sym6_316	123	98	522	343	738	396	392	**254**	**337**	259	0.86	1.02
4gt12-v0_86	116	98	371	284	820	460	283	**227**	**258**	231	0.91	1.02
4gt4-v0_72	113	95	340	265	858	466	295	**220**	**251**	224	0.85	1.02
4gt4-v0_78	109	99	353	271	713	447	328	237	**243**	**229**	0.74	0.97
mod10_171	108	97	323	272	753	425	301	**226**	**240**	229	0.80	1.01
4gt4-v1_74	119	108	375	312	937	522	367	266	**278**	**258**	0.76	0.97
rd53_135	134	114	434	338	918	507	424	**290**	**323**	297	0.76	1.02
mini-alu_167	126	111	357	311	925	536	338	**262**	**282**	264	0.83	1.01
one-two-three-v0_97	128	116	397	335	812	474	380	281	**287**	**270**	0.76	0.96
ham7_104	149	134	506	393	954	562	397	316	**343**	**312**	0.86	0.99
decod24-enable_126	149	134	472	397	908	553	428	323	**341**	**322**	0.80	1.00
mod8-10_178	152	135	510	400	1030	605	449	331	**338**	**315**	0.75	0.95
cnt3-5_180	215	148	683	451	1327	691	**585**	**390**	601	461	1.03	1.18
ex3_229	175	157	588	470	1269	718	539	391	**403**	**371**	0.75	0.95
4gt4-v0_73	179	160	535	445	1180	690	470	**370**	**401**	376	0.85	1.02
mod8-10_177	196	178	573	498	1354	775	563	424	**448**	**418**	0.80	0.99
C17_204	205	173	673	523	1892	891	614	**447**	**502**	454	0.82	1.02
alu-v2_31	198	172	564	476	1350	816	547	**406**	**438**	413	0.80	1.02
rd53_131	200	175	701	554	1230	749	624	**421**	**474**	432	0.76	1.03
alu-v2_30	223	199	664	546	1595	878	682	491	**502**	**472**	0.76	0.96
mod5adder_127	239	208	793	633	1705	1022	702	528	**533**	**499**	0.76	0.95
rd53_133	256	221	805	640	1703	977	739	**543**	**615**	554	0.83	1.02
cm82a_208	283	234	1003	724	2092	1081	848	581	**665**	**576**	0.78	0.99
majority_239	267	232	932	704	1985	1102	805	581	**624**	**568**	0.78	0.98
ex2_227	275	241	965	737	1907	1107	789	**584**	**656**	603	0.83	1.03
sf_276	336	301	1104	902	2083	1290	935	743	**762**	**727**	0.81	0.98
sf_274	336	300	1095	878	2145	1287	922	**686**	**757**	725	0.82	1.06
con1_216	415	346	1454	1078	3593	1872	1288	**891**	**1075**	940	0.83	1.05
wim_266	427	352	1432	1058	3482	1823	1209	**870**	**1089**	951	0.90	1.09
rd53_130	448	383	1430	1132	3563	1905	1300	**930**	**1099**	976	0.85	1.05
f2_232	525	449	1796	1364	4409	2187	1563	**1154**	**1309**	1205	0.84	1.04
cm152a_212	532	461	1813	1374	5455	2557	1510	**1078**	**1364**	1221	0.90	1.13
rd53_251	564	492	1851	1474	5391	2555	1629	**1221**	**1405**	1268	0.86	1.04
hwb5_53	598	535	1850	1525	6201	2717	1713	1320	**1360**	**1276**	0.79	0.97
cm42a_207	771	634	2607	1926	8818	3910	2284	**1600**	**1920**	1691	0.84	1.06

			Qiskit 0.7.0		Project Q 0.4.1		Quilc 1.1.1 Pyquil 2.1.1		CQC's t|ket⟩		t|ket⟩ comparison	
Name	g_{in}	d_{in}	g_{out}	d_{out}	g_{out}	d_{out}	g_{out}	d_{out}	g_{out}	d_{out}	r_{gate}	r_{depth}
pm1_249	771	634	2713	2043	8818	3910	2220	**1555**	**1920**	1691	0.86	1.09
dc1_220	833	705	2895	2194	11741	4784	2351	**1690**	**2077**	1770	0.88	1.05
squar5_261	869	720	3015	2189	10135	4466	2676	**1845**	**2358**	1954	0.88	1.06
z4_268	1343	1112	4744	3473	22664	8032	4104	**2870**	**3583**	3059	0.87	1.07
sqrt8_260	1314	1121	4853	3633	22130	8209	4107	**2937**	**3416**	2985	0.83	1.02
radd_250	1405	1210	4952	3782	23920	8909	4290	**3027**	**3768**	3281	0.88	1.08
adr4_197	1498	1249	5284	3883	22303	8649	4538	**3142**	**3871**	3279	0.85	1.04
sym6_145	1701	1499	5356	4324	40194	11879	4910	**3675**	**4170**	3820	0.85	1.04
misex1_241	2100	1797	–	–	20756	10450	5798	**4367**	**5578**	4941	0.96	1.13
rd73_252	2319	1963	–	–	45433	15172	–	–	**6007**	**5173**	0.13	0.34
cycle10_2_110	2648	2276	–	–	50777	17976	–	–	**7084**	**6316**	0.14	0.35
hwb6_56	2952	2559	–	–	72955	21358	–	–	**7125**	**6496**	0.10	0.30
square_root_7	3089	2520	–	–	37275	16316	–	–	**8928**	**7401**	0.24	0.45
ham15_107	3858	3273	–	–	46860	21421	–	–	**10206**	**8884**	0.22	0.41
dc2_222	4131	3518	–	–	45200	21119	–	–	**11545**	**10210**	0.26	0.48
sqn_258	4459	3719	–	–	93394	29983	–	–	**11425**	**9743**	0.12	0.32
inc_237	4636	3928	–	–	39432	21180	–	–	**12381**	**10872**	0.31	0.51
cm85a_209	4986	4256	–	–	71744	29352	–	–	**13297**	**11722**	0.19	0.40
rd84_253	5960	4917	–	–	118473	40259	–	–	**16027**	**13432**	0.14	0.33
co14_215	7840	5759	–	–	84431	36897	–	–	**22357**	**17144**	0.26	0.46
root_255	7493	5965	–	–	128324	47123	–	–	**20332**	**16951**	0.16	0.36
mlp4_245	8232	6930	–	–	50107	31076	–	–	**22759**	**19640**	0.45	0.63
urf2_277	10066	8312	–	–	243218	71917	–	–	**25127**	**21186**	0.10	0.29
sym9_148	9408	8062	–	–	241725	73636	–	–	**23489**	**20883**	0.10	0.28
life_238	9800	8356	–	–	221220	72011	–	–	**26064**	**22788**	0.12	0.32
hwb7_59	10681	9112	–	–	264057	78857	–	–	**26033**	**23163**	0.10	0.29
max46_240	11844	9657	–	–	269243	84007	–	–	**30709**	**25891**	0.11	0.31
clip_206	14772	12028	–	–	213901	85381	–	–	**40504**	**34045**	0.19	0.40
9symml_195	15232	12849	–	–	–	–	–	–	**40897**	**35349**		
sym9_193	15232	12849	–	–	–	–	–	–	**40897**	**35349**		
sao2_257	16864	13209	–	–	–	–	–	–	**46536**	**37963**		
dist_223	16624	13274	–	–	–	–	–	–	**45230**	**37762**		
urf5_280	23764	19888	–	–	–	–	–	–	**60233**	**51479**		
urf1_278	26692	22307	–	–	–	–	–	–	**69370**	**58597**		
sym10_262	28084	23736	–	–	–	–	–	–	**76180**	**65905**		
hwb8_113	30372	26041	–	–	–	–	–	–	**77247**	**68560**		
urf2_152	35210	32247	–	–	–	–	–	–	**84345**	**77320**		
urf3_279	60380	50568	–	–	–	–	–	–	**158115**	**132770**		
plus63mod4096_16	356329	48265	–	–	–	–	–	–	**155209**	**135715**		
urf5_158	71932	64750	–	–	–	–	–	–	**177757**	**158774**		
urf6_160	75180	67637	–	–	–	–	–	–	**209231**	**174548**		
urf1_149	80878	72933	–	–	–	–	–	–	**200949**	**179071**		
plus63mod8192_16	481865	70222	–	–	–	–	–	–	**226354**	**198162**		
hwb9_119	90955	77968	–	–	–	–	–	–	**233049**	**205682**		
urf3_155	185276	167215	–	–	–	–	–	–	**463538**	**409727**		

Name	g_{in}	d_{in}	Qiskit 0.7.0 g_{out}	Qiskit 0.7.0 d_{out}	Project Q 0.4.1 g_{out}	Project Q 0.4.1 d_{out}	Quilc 1.1.1 Pyquil 2.1.1 g_{out}	Quilc 1.1.1 Pyquil 2.1.1 d_{out}	CQC's t|ket⟩ g_{out}	CQC's t|ket⟩ d_{out}	t|ket⟩ comparison r_{gate}	t|ket⟩ comparison r_{depth}
ground_state _es-timation_10	154209	151095	–	–	–	–	–	–	**5039**	**3897**		
urf4_187	224028	185975	–	–	–	–	–	–	**568938**	**481368**		

g: the number of quantum gates (elementary operations),
d: depth of the quantum circuits and – are time-outs

B.3 CX Only Comparison on IBM Tokyo

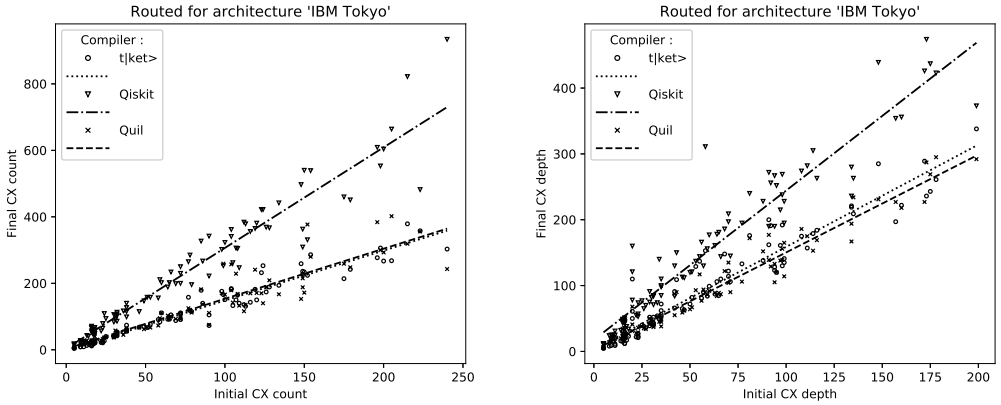


Figure 18 Routing comparison on IBM Tokyo, CX count and CX depth when counting only CX gates. The charts are a zoomed in version of the initial segment of lower charts of Fig. 14.

Table 6 CX gates only comparison on IBM Tokyo.

Name	g_{in}	d_{in}	Qiskit 0.7.0 g_{out}	Qiskit 0.7.0 d_{out}	Quilc 1.1.1 Pyquil 2.1.1 g_{out}	Quilc 1.1.1 Pyquil 2.1.1 d_{out}	CQC's t|ket⟩ g_{out}	CQC's t|ket⟩ d_{out}	t|ket⟩ comparison r_{gate}	t|ket⟩ comparison r_{depth}
xor5_254	5	5	14	11	7	7	**5**	**5**	0.71	0.71
graycode6_47	5	5	17	11	**5**	**5**	**5**	**5**	1.00	1.00
ex1_226	5	5	18	12	7	7	**5**	**5**	0.71	0.71
4gt11_84	9	8	27	24	15	14	**9**	**8**	0.60	0.57
4mod5-v0_20	10	9	25	19	**16**	**15**	19	18	1.19	1.20
ex-1_166	9	9	28	25	18	15	**9**	**9**	0.50	0.60
4mod5-v1_22	11	10	26	25	**20**	**18**	23	22	1.15	1.22
mod5d1_63	13	11	43	31	28	23	**13**	**11**	0.46	0.48
ham3_102	11	11	25	25	18	15	**9**	**9**	0.50	0.60
4gt11_83	14	14	41	32	20	20	**17**	**16**	0.85	0.80
4gt11_82	18	18	49	34	30	27	**24**	**23**	0.80	0.85
rd32-v0_66	16	16	57	43	21	20	**12**	**12**	0.57	0.60
alu-v0_27	17	15	63	48	53	34	**20**	**19**	0.38	0.56
4mod5-v1_24	16	15	66	49	28	27	**16**	**15**	0.57	0.56
4mod5-v0_19	16	16	67	51	**25**	**25**	**25**	**25**	1.00	1.00

			Qiskit 0.7.0		Quilc 1.1.1 Pyquil 2.1.1		CQC's t|ket⟩		t|ket⟩ comparison	
Name	g_{in}	d_{in}	g_{out}	d_{out}	g_{out}	d_{out}	g_{out}	d_{out}	r_{gate}	r_{depth}
mod5mils_65	16	16	71	48	28	28	**25**	**25**	0.89	0.89
rd32-v1_68	16	16	53	45	24	17	**12**	**12**	0.50	0.71
alu-v1_28	18	16	60	35	40	26	**21**	**20**	0.53	0.77
alu-v2_33	17	15	59	44	29	28	**20**	**19**	0.69	0.68
alu-v4_37	18	16	67	45	52	33	**21**	**20**	0.40	0.61
alu-v3_35	18	16	57	43	52	33	**21**	**20**	0.40	0.61
3_17_13	17	17	46	38	26	25	**17**	**17**	0.65	0.68
alu-v1_29	17	15	57	43	53	34	**20**	**19**	0.38	0.56
miller_11	23	23	65	53	32	29	**23**	**23**	0.72	0.79
alu-v3_34	24	23	109	74	37	36	**27**	**27**	0.73	0.75
decod24-v2_43	22	22	57	55	28	28	**22**	**22**	0.79	0.79
decod24-v0_38	23	23	79	57	27	25	**21**	**21**	0.78	0.84
mod5d2_64	25	25	94	77	**38**	**38**	40	40	1.05	1.05
4gt13_92	30	26	84	65	**39**	**35**	42	38	1.08	1.09
4gt13-v1_93	30	27	106	70	**37**	**34**	39	36	1.05	1.06
4mod5-v0_18	31	31	91	71	58	46	**43**	**40**	0.74	0.87
decod24-bdd_294	32	31	102	85	**44**	**37**	53	52	1.20	1.41
one-two-three-v2_100	32	29	86	75	**46**	**41**	53	52	1.15	1.27
one-two-three-v3_101	32	29	108	91	53	50	**42**	**39**	0.79	0.78
4mod5-v1_23	32	30	113	82	50	49	**47**	**42**	0.94	0.86
4mod5-bdd_287	31	31	103	84	46	**39**	**43**	43	0.93	1.10
rd32_270	36	35	117	98	**45**	**38**	54	53	1.20	1.39
4gt5_75	38	33	99	74	**56**	**53**	**56**	54	1.00	1.02
alu-bdd_288	38	35	140	110	**65**	**45**	74	72	1.14	1.60
alu-v0_26	38	35	115	85	**56**	**48**	57	56	1.02	1.17
decod24-v1_41	38	35	127	100	**56**	**50**	62	62	1.11	1.24
rd53_138	60	42	202	130	**111**	**87**	**111**	91	1.00	1.05
4gt5_76	46	42	115	84	**67**	**55**	68	66	1.01	1.20
4gt13_91	49	46	147	113	**62**	**59**	70	65	1.13	1.10
cnt3-5_179	85	43	327	150	182	**90**	**179**	109	0.98	1.21
qft_10	90	34	342	140	75	**45**	**73**	50	0.97	1.11
4gt13_90	53	50	158	120	**67**	**64**	77	72	1.15	1.12
alu-v4_36	51	47	158	112	**67**	**62**	**67**	63	1.00	1.02
mini_alu_305	77	53	265	137	**125**	**93**	158	129	1.26	1.39
ising_model_10	90	20	222	78	111	**41**	**105**	50	0.95	1.22
ising_model_16	150	20	540	160	**172**	**31**	234	110	1.36	3.55
ising_model_13	120	20	381	121	171	68	**150**	**61**	0.88	0.90
4gt5_77	58	51	156	121	**73**	**67**	91	82	1.25	1.22
sys6-v0_111	98	55	302	144	**170**	**117**	176	137	1.04	1.17
one-two-three-v1_99	59	56	211	166	95	**77**	**84**	81	0.88	1.05
one-two-three-v0_98	65	59	202	156	94	86	**92**	**84**	0.98	0.98
decod24-v3_45	64	57	189	130	103	94	**92**	**90**	0.89	0.96
4gt10-v1_81	66	60	234	177	**100**	**92**	105	99	1.05	1.08
aj-e11_165	69	63	199	161	93	89	**88**	**87**	0.95	0.98

			Qiskit 0.7.0		Quilc 1.1.1 Pyquil 2.1.1		CQC's t|ket⟩		t|ket⟩ comparison	
Name	g_{in}	d_{in}	g_{out}	d_{out}	g_{out}	d_{out}	g_{out}	d_{out}	r_{gate}	r_{depth}
4mod7-v0_94	72	66	200	144	**97**	**87**	111	110	1.14	1.26
alu-v2_32	72	64	250	178	**92**	**84**	109	106	1.18	1.26
rd73_140	104	68	341	186	**179**	**122**	182	148	1.02	1.21
4mod7-v1_96	72	65	230	181	100	89	**91**	**85**	0.91	0.96
4gt4-v0_80	79	71	196	145	**112**	**90**	115	106	1.03	1.18
mod10_176	78	70	285	197	120	**104**	**118**	114	0.98	1.10
0410184_169	104	70	362	209	257	135	**184**	**122**	0.72	0.90
qft_16	240	58	934	311	**243**	**90**	303	153	1.25	1.70
4gt12-v0_88	86	77	267	195	**137**	**106**	140	133	1.02	1.25
rd84_142	154	81	539	240	**280**	**172**	286	176	1.02	1.02
rd53_311	124	92	422	256	–	–	**253**	**191**	0.60	0.75
4_49_16	99	91	253	190	**140**	**129**	166	162	1.19	1.26
sym9_146	148	91	497	272	**259**	**177**	**259**	200	1.00	1.13
4gt12-v1_89	100	88	323	228	153	139	**151**	**138**	0.99	0.99
4gt12-v0_87	112	94	384	267	166	152	**136**	**123**	0.82	0.81
4gt4-v0_79	105	94	262	190	**116**	**105**	133	128	1.15	1.22
hwb4_49	107	99	306	228	**142**	**133**	150	141	1.06	1.06
sym6_316	123	98	422	269	**229**	**165**	232	192	1.01	1.16
4gt12-v0_86	116	98	356	238	170	156	**143**	**130**	0.84	0.83
4gt4-v0_72	113	95	379	252	170	139	**131**	**120**	0.77	0.86
4gt4-v0_78	109	99	247	195	**135**	**120**	140	135	1.04	1.12
mod10_171	108	97	305	215	**133**	**114**	165	161	1.24	1.41
4gt4-v1_74	119	108	365	274	**168**	**138**	182	175	1.08	1.27
rd53_135	134	114	442	305	241	186	**194**	**179**	0.80	0.96
mini-alu_167	126	111	369	282	**167**	**153**	174	157	1.04	1.03
one-two-three-v0_97	128	116	367	263	**188**	**175**	189	184	1.01	1.05
ham7_104	149	134	309	236	**209**	**169**	236	221	1.13	1.31
decod24-enable_126	149	134	363	280	**215**	**194**	227	219	1.06	1.13
mod8-10_178	152	135	331	263	**188**	**167**	224	209	1.19	1.25
cnt3-5_180	215	148	822	439	**377**	**234**	379	285	1.01	1.22
ex3_229	175	157	460	354	319	231	**214**	**197**	0.67	0.85
4gt4-v0_73	179	160	451	356	259	227	**242**	**222**	0.93	0.98
mod8-10_177	196	178	609	423	**248**	**218**	277	261	1.12	1.20
C17_204	205	173	664	474	384	295	**268**	**236**	0.70	0.80
alu-v2_31	198	172	553	426	402	**287**	**306**	289	0.76	1.01
rd53_131	200	175	604	437	300	**227**	**267**	243	0.89	1.07
alu-v2_30	223	199	482	373	**293**	**269**	358	338	1.22	1.26
mod5adder_127	239	208	650	476	356	292	**311**	**288**	0.87	0.99
rd53_133	256	221	703	499	359	**308**	**348**	328	0.97	1.06
cm82a_208	283	234	914	607	**393**	**317**	451	391	1.15	1.23
majority_239	267	232	720	528	402	328	**348**	**311**	0.87	0.95
ex2_227	275	241	886	648	443	**371**	**416**	391	0.94	1.05
sf_276	336	301	989	731	**372**	**340**	489	453	1.31	1.33
sf_274	336	300	863	643	457	391	**357**	**328**	0.78	0.84
con1_216	415	346	1389	916	**659**	**486**	736	626	1.12	1.29
wim_266	427	352	1145	793	**763**	**501**	772	688	1.01	1.37

			Qiskit 0.7.0		Quilc 1.1.1 Pyquil 2.1.1		CQC's t|ket⟩		t|ket⟩ comparison	
Name	g_{in}	d_{in}	g_{out}	d_{out}	g_{out}	d_{out}	g_{out}	d_{out}	r_{gate}	r_{depth}
rd53_130	448	383	1242	899	695	**547**	**646**	608	0.93	1.11
f2_232	525	449	1443	1069	**722**	**621**	854	772	1.18	1.24
cm152a_212	532	461	1491	1051	805	**633**	**793**	717	0.99	1.13
rd53_251	564	492	1487	1058	960	**659**	**805**	715	0.84	1.08
hwb5_53	598	535	1568	1179	941	765	**808**	**757**	0.86	0.99
cm42a_207	771	634	2333	1593	1411	996	**1102**	**968**	0.78	0.97
pm1_249	771	634	2303	1574	1411	996	**1102**	**968**	0.78	0.97
dc1_220	833	705	2450	1715	1532	**1049**	**1436**	1261	0.94	1.20
squar5_261	869	720	2902	1948	**1463**	**1063**	1631	1372	1.11	1.29
z4_268	1343	1112	4185	2828	2282	**1632**	**2235**	1914	0.98	1.17
sqrt8_260	1314	1121	4079	2846	2406	**1665**	**2235**	1964	0.93	1.18
radd_250	1405	1210	4346	3050	–	–	**2697**	**2361**	0.62	0.77
adr4_197	1498	1249	4584	3110	**2333**	**1720**	2737	2368	1.17	1.38
sym6_145	1701	1499	4872	3517	2554	**2042**	**2301**	2145	0.90	1.05
misex1_241	2100	1797	–	–	**3249**	**2435**	4291	3760	1.32	1.54
rd73_252	2319	1963	–	–	3854	**2798**	**3748**	3276	0.97	1.17
cycle10_2_110	2648	2276	–	–	4511	**3463**	**4342**	3802	0.96	1.10
hwb6_56	2952	2559	–	–	4405	**3575**	**4205**	3904	0.95	1.09
square_root_7	3089	2520	–	–	**4967**	**3427**	5790	4833	1.17	1.41
ham15_107	3858	3273	–	–	6649	**4828**	**6485**	5605	0.98	1.16
dc2_222	4131	3518	–	–	–	–	**7694**	**6708**		
sqn_258	4459	3719	–	–	–	–	**6863**	**5984**		
inc_237	4636	3928	–	–	–	–	**8274**	**7176**		
cm85a_209	4986	4256	–	–	–	–	**8166**	**7093**		
rd84_253	5960	4917	–	–	–	–	**9506**	**8154**		
co14_215	7840	5759	–	–	–	–	**14415**	**11195**		
root_255	7493	5965	–	–	–	–	**11971**	**9940**		
mlp4_245	8232	6930	–	–	–	–	**15156**	**13061**		
urf2_277	10066	8312	–	–	–	–	**17367**	**15384**		
sym9_148	9408	8062	–	–	–	–	**13893**	**12321**		
life_238	9800	8356	–	–	–	–	**15944**	**13926**		
hwb7_59	10681	9112	–	–	–	–	**17655**	**15828**		
max46_240	11844	9657	–	–	–	–	**19519**	**16979**		
clip_206	14772	12028	–	–	–	–	**25499**	**21345**		
9symml_195	15232	12849	–	–	–	–	**24532**	**21295**		
sym9_193	15232	12849	–	–	–	–	**24532**	**21295**		
sao2_257	16864	13209	–	–	–	–	**28799**	**23847**		
dist_223	16624	13274	–	–	–	–	**27953**	**23342**		
urf5_280	23764	19888	–	–	–	–	**41610**	**36329**		
urf1_278	26692	22307	–	–	–	–	**46023**	**40020**		
sym10_262	28084	23736	–	–	–	–	**44383**	**38132**		
hwb8_113	30372	26041	–	–	–	–	**50008**	**44245**		
urf2_152	35210	32247	–	–	–	–	**58307**	**53905**		
urf3_279	60380	50568	–	–	–	–	**105870**	**91838**		
plus63mod4096_163	56329	48265	–	–	–	–	**95107**	**82375**		
urf5_158	71932	64750	–	–	–	–	**122288**	**110318**		

			Qiskit 0.7.0		Quilc 1.1.1 Pyquil 2.1.1		CQC's t|ket⟩		t|ket⟩ comparison	
Name	g_{in}	d_{in}	g_{out}	d_{out}	g_{out}	d_{out}	g_{out}	d_{out}	r_{gate}	r_{depth}
urf6_160	75180	67637	–	–	–	–	**141416**	**120999**		
urf1_149	80878	72933	–	–	–	–	**137791**	**123873**		
plus63mod8192_164	81865	70222	–	–	–	–	**145872**	**125957**		
hwb9_119	90955	77968	–	–	–	–	**149106**	**131678**		
urf3_155	185276	167215	–	–	–	–	**317732**	**285000**		
ground_state _estimation_10	154209	151095	–	–	–	–	**1015**	**776**		
urf4_187	224028	185975	–	–	–	–	**385355**	**330368**		

g: the number of quantum gates (elementary operations),
d: depth of the quantum circuits and – are time-outs.

Applications of the Quantum Algorithm for st-Connectivity

Kai DeLorenzo
Middlebury College, Computer Science Department, Middlebury, VT, USA

Shelby Kimmel
Middlebury College, Computer Science Department, Middlebury, VT, USA

R. Teal Witter
Middlebury College, Computer Science Department, Middlebury, VT, USA

Abstract

We present quantum algorithms for various problems related to graph connectivity. We give simple and query-optimal algorithms for cycle detection and odd-length cycle detection (bipartiteness) using a reduction to st-connectivity. Furthermore, we show that our algorithm for cycle detection has improved performance under the promise of large circuit rank or a small number of edges. We also provide algorithms for detecting even-length cycles and for estimating the circuit rank of a graph. All of our algorithms have logarithmic space complexity.

2012 ACM Subject Classification Theory of computation → Quantum query complexity; Theory of computation → Graph algorithms analysis

Keywords and phrases graphs, algorithms, query complexity, quantum algorithms, span programs

Digital Object Identifier 10.4230/LIPIcs.TQC.2019.6

Funding This research was sponsored by the Army Research Office and was accomplished under Grant Number W911NF-18-1-0286. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Army Research Office or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation herein.

Acknowledgements We thank Chris Cade and Stacey Jeffery for helpful discussions.

1 Introduction

Quantum query algorithms are remarkably described by span programs [15, 16], a linear algebraic object originally created to study classical logspace complexity [12]. However, finding optimal span program algorithms can be challenging; while they can be obtained using a semidefinite program, the size of the program grows exponentially with the size of the input to the algorithm. Moreover, span programs are designed to characterize the query complexity of an algorithm, while in practice we also care about the time and space complexity.

One of the nicest span programs is for the problem of undirected *st*-connectivity, in which one must decide whether two vertices s and t are connected in a given graph. It is "nice" for several reasons:

- It is easy to describe and understand why it is correct.
- It corresponds to a quantum algorithm that uses logarithmic (in the number of vertices and edges of the graph) space [4, 11].
- The time complexity of the corresponding algorithm is the product of the query complexity, and the time required to implement a unitary that applies one step of a quantum walk on the underlying graph [4, 11]. On the complete graph, for example, the quantum walk step introduces only an additional logarithmic factor to the complexity [4].

- The query complexity of the algorithm is determined by two well known graph functions, the effective resistance and effective capacitance [10].

Thus one strategy for designing other "nice" quantum algorithms is to reduce a given problem to *st*-connectivity, and then use the span program *st*-connectivity algorithm. This strategy has proven to be quite successful, and in fact has produced several optimal or nearly optimal algorithms. There is a reduction from Boolean formula evaluation to *st*-connectivity [14] that produces an optimal quantum algorithm for read-once formulas [11]. There is an optimal reduction from graph connectivity to *st*-connectivity [10]. Cade et al. use an *st*-connectivity subroutine to create nearly query-optimal algorithms for cycle detection and bipartiteness [7][1]. Finally, the *st*-connectivity span program algorithm underlies the learning graph framework [3], one of the most successful heuristics for span program algorithm design.

In this work, we follow precisely this strategy for creating "nice" quantum algorithms: we reduce the graph problems of cycle detection, odd-length path detection, bipartiteness, and even-length cycle detection to *st*-connectivity. In our reductions, solving the related *st*-connectivity problem comprises the whole algorithm; in other words, we create a new graph that has an *st*-path if and only if the original graph has the property in question.

Additionally, there is an an estimation algorithm closely related to the *st*-connectivity algorithm that determines the size of the effective resistance or effective capacitance of the graph [9, 10]. Not only is it often useful to estimate the effective resistance or effective capacitance of a graph, as these quantities bound the shortest path length and smallest cut size respectively, but sometimes one can encode quantities of interest as either the effective capacitance or effective resistance. For example, given a graph G, it is possible to create a new graph whose effective resistance is the average effective resistance (Kirchoff index) of the original graph [10]. Using this strategy of reduction to effective resistance estimation, we also create an algorithm to estimate the circuit rank of a graph.

1.1 Contributions and Comparison to Previous Work

All of our algorithms are in the adjacency matrix model (see Section 2), in which one can query elements of the adjacency matrix of the input graph. This contrasts with work such as [7] which study similar problems in the adjacency list model.

We note all of our algorithms are space efficient, in that the number of qubits required are logarithmic in the number of edges and vertices of the graph. (This property is inherited directly from the basic *st*-connectivity span program.) We do not analyze time complexity, but, as mentioned above, it is the product of the query complexity, which we analyze, and the time required to implement certain quantum walk unitaries. (We leave this analysis for future work.)

We next discuss the context of each of our results in turn. In this section, we assume the underlying graph is the complete graph on n vertices to more easily compare to previous work, although in the main body of the paper, we show our results apply more to generic underlying graphs with n vertices and m edges.

Cycle Detection

Cade et al. [7] describe a nearly optimal $\tilde{O}(n^{3/2})$ quantum query algorithm for cycle detection via reduction to *st*-connectivity, almost matching the lower bound of $\Omega(n^{3/2})$ [8]. In this work, we find an algorithm that removes the log-factors of the previous result, giving an

[1] In Ref. [2], Āriņš designed algorithms for connectivity and bipartiteness which, while not strictly reductions to *st*-connectivity, are very closely related.

algorithm with optimal $O(n^{3/2})$ query complexity. Moreover, our algorithm is simpler than that in Ref. [7]: their approach requires solving an *st*-connectivity problem within a Grover search, while our approach is entirely based on solving an *st*-connectivity problem.

We furthermore prove that if promised that (in the case of a cycle) the circuit rank of the graph is at least r or (in the case of no cycles) there are at most μ edges, then the query complexity of cycle detection is $O(\mu\sqrt{n/r})$.

Bipartiteness

An optimal quantum query algorithm for bipartiteness was created by Ārinš [2], matching the lower bound of $\Omega(n^{3/2})$ [19]. However, this algorithm was not known to be time efficient (and did not use a reduction to *st*-connectivity, although the ideas are quite similar to the approach here and in [7]). In Ref. [7], Cade et al., using similar ideas as in their cycle detection algorithm, create a bipartiteness checking algorithm using a reduction to *st*-connectivity that is again embedded in a search loop, which is optimal up to logarithmic factors in query complexity. Our algorithm for bipartiteness removes the logarithmic factors of [7], and so recovers the optimal query complexity of [2], while retaining the simplicity of the reduction to *st*-connectivity.

Even-length Cycle Detection

The problem of detecting an even-length cycle can provide insight into the structure of the graph. For example, it is straightforward to see that in a graph with no even cycles, no edge can be involved in more than one cycle. Classically this problem requires $\Theta(n^2)$ queries [17]. We are not aware of an existing quantum algorithm for this problem; we provide an $O(n^{3/2})$ query algorithm.

Estimation of Circuit Rank

Circuit rank parameterizes the number of cycles in a graph: it is the number of edges that must be removed before there are no cycles left in a graph. It has also been used to describe the complexity of computer programs [13]. We give an algorithm to estimate the circuit rank r to multiplicative error ϵ with query complexity $\widetilde{O}\left(\epsilon^{-3/2}\sqrt{n^4/r}\right)$ in the generic case and query complexity $\widetilde{O}\left(\epsilon^{-3/2}\sqrt{n^3/r}\right)$ when promised that the graph is a cactus graph. When additionally promised that the circuit rank is large, these algorithms can have non-trivial query complexity. We are aware of no other classical or quantum query algorithms that determine or estimate this quantity.

Odd-length Path

We provide an algorithm to determine whether there is an odd-length path between two specified vertices that uses $O(n^{3/2})$ queries. While perhaps not the most interesting problem on its own, we effectively leverage this algorithm as a subroutine in several of our other constructions.

1.2 Open Problems

Throughout this paper, our strategy is to take a graph G, use it to create a new graph G', such that there is an *st*-path through G' if and only if G has a certain property. We analyze the query complexity of the algorithms we create in detail, but not the time complexity. The

time complexities of our algorithms depend on the time required to implement one step of a quantum walk on the graph G' (see U from Theorem 4 and Ref. [11]) [2]. We strongly suspect that the highly structured nature of the graphs G' we consider would yield time-efficient algorithms, but we have not done a full analysis.

In the case of our algorithm for cycle detection, we create a graph G' whose effective resistance is the circuit rank of the original graph G. For the graphs we design for the other algorithms in this paper, do the effective resistance or effective capacitance have relevant meanings?

The query complexity of our algorithm for estimating the circuit rank of a graph depends on bounding a quantity called the approximate negative witness. While the best bound we currently have is $O(n^4)$, we believe this is not tight. Obtaining a better bound would not only be interesting for these results, but could provide insight into more general quantum estimation algorithms.

Several of our results rely on a graph that tests for paths of odd length, i.e. those whose length modulo 2 is 1. Is there a way to adapt our algorithm to test for paths of arbitrary modulus?

Ref. [1] provides a list of complete problems for symmetric logarithmic space (SL), of which *st*-connectivity is one such problem. It would be interesting to study the query complexity of these problems to see if reducing to *st*-connectivity always gives an optimal approach.

Finally, it would be nice to improve the quantum lower bounds and classical bounds for several of these problems. In particular, we would like to obtain better quantum lower bounds for the even-length cycle detection and odd-length path detection problems. (For the later, the best quantum lower bound is $\Omega(n)$ [4].) We expect that the promise of large circuit rank also aids a classical algorithm for cycle detection, and it would be interesting to know by how much, in order to compare to our quantum algorithm.

2 Preliminaries

We consider undirected graphs $G = (V, E)$ where V is a set of vertices and E a set of edges; we often use $E(G)$ and $V(G)$ to denote the sets of edges and vertices of a graph G when there are multiple graphs involved. If clear which graph we are referring to, we will use n for the number of vertices in the graph, and m for the number of edges. For ease of notation, we associate each edge with a unique label ℓ. For example, we refer to an edge between vertices u and v labeled by ℓ as $\{u, v\}$, or simply as ℓ. In general, we could consider a weighting function on the edges or consider graphs with multi-edges (see [10] for more details on how these modifications are implemented) but for our purposes, we will always consider all edges to have weight 1, and we will only consider graphs with a at most a single edge between any two vertices.

We will use the following notation regarding spanning trees: if G is a connected graph and $\ell \in E(G)$, then we use $t_\ell(G)$ to denote the number of unique spanning trees of G that include edge ℓ, and we use $t(G)$ to denote the total number of unique spanning trees of G.

In Section 3 we describe a quantum algorithm for estimating the circuit rank of a graph, which is a quantity that is relevant for a number of applications, like determining the robustness of a network, analyzing chemical structure [18], or parameterizing the complexity of a program [13].

[2] In Ref. [7], they claim that their algorithms are time efficient, but their time analysis only considers the case of the underlying graph being the complete graph, while the actual graph used in their algorithms is not the complete graph. We expect that their algorithms can be implemented efficiently, but more work is needed to show this.

▶ **Definition 1** (Circuit Rank). *The* circuit rank *of a graph with m edges and n vertices is $m - n + \kappa$ where κ is the number of connected components. Alternatively, the circuit rank of a graph is the minimum number of edges that must be removed to break all cycles and turn the graph into a tree or forest.*

We also consider a special class of graphs called cactus graphs:

▶ **Definition 2.** *A cactus graph is a connected graph in which any two simple cycles share at most one common vertex.*

We will in particular use cactus forests, in which all components of the graph are cacti. For cactus forests, the circuit rank is simply the total number of cycles in the graph.

The final type of graph we need is the bipartite double graph:

▶ **Definition 3.** *Given a graph G, the bipartite double of G, denoted K^G, is the graph that consists of two copies of the vertices of G (with vertex $v \in V(G)$ labeled as v_0 in the first copy and v_1 in the second), with all original edges removed, and edges $\{u_0, v_1\}$ and $\{v_0, u_1\}$ created for each edge $\{u, v\} \in E(G)$. This graph is also known as the Kronecker cover of G, or $G \times K_2$, and its adjacency matrix is given by $G \otimes X$ (where X is the Pauli X operator.)*

We associate edges of G with literals of a string $x \in \{0,1\}^N$, where $N \leq |E|$, and a literal is either x_i or $\overline{x}_i$ for $i \in [N]$. The subgraph $G(x)$ of G contains an edge ℓ if ℓ is associated with x_i and $x_i = 1$, or if ℓ is associated with $\overline{x}_i$ and $x_i = 0$. (See [10] for more details on this association.)

We assume that we have complete knowledge of G, and access to $x \in \{0,1\}^N$ via a black box unitary (oracle) O_x. This oracle acts as $O_x|i\rangle|b\rangle = |i\rangle|b \oplus x_i\rangle$, where x_i is the i^{th} bit of x. Then our goal is to use O_x as few times as possible to determine a property of the graph $G(x)$. The number of uses of O_x required for a given application is called the query complexity.

We will be applying and analyzing an algorithm for st-connectivity, which is the problem of deciding whether two nodes $s, t \in V(G)$ are connected in a graph $G(x)$, where G is initially known, but x must be determined using the oracle, and we are promised $x \in X$, where $X \subseteq \{0,1\}^N$. An st-path is a series of edges connecting s to t.

Given two instances of st-connectivity, they can be combined in *parallel*, where the two s vertices are identified and labeled as the new s, and the two t vertices are identified and labeled as the new t. This new st-connectivity problem encodes the logical OR of the original connectivity problems, in that the new graph is connected if and only if at least one of the original graphs was connected. Two instances of st-connectivity can also be combined in *series*, where the s vertex of one graph is identified with the t vertex of the other graph, and relabeled using a label not previously used for a vertex in either graph. This new st-connectivity encodes the logical AND of the original connectivity problems, in that the new graph is connected if and only if both of the original graphs were connected. This correspondence was noted in [14] and applied to quantum query algorithms for Boolean formulas in [11] and for determining total connectivity in [10].

The key figures of merit for determining the query complexity of the span-program-based st-connectivity quantum algorithm are effective resistance and effective capacitance [10]. We use $R_{s,t}(G(x))$ to denote the effective resistance between vertices s and t in a graph $G(x)$, and we use $C_{s,t}(G(x))$ to denote the effective capacitance between vertices s and t in $G(x)$. These two functions were originally formulated to characterize electrical circuits, but are also important functions in graph theory. For example, effective resistance is related to the hitting time of a random walk on a graph. (See [6] for more information on effective resistance and capacitance in the context of graph theory.) For this paper, we don't require a formal definition of these functions, but can instead use a few of their well known and easily derived properties (see [10] for formal definitions):

Effective Resistance

1. If G consists of a single edge between s and t, and $G(x) = G$, then $R_{s,t}(G(x)) = 1$.
2. If s and t are not connected in $G(x)$, then $R_{s,t}(G(x)) = \infty$.
3. If G consists of subgraphs G_1 and G_2 connected in series as described above, then $R_{s,t}(G(x)) = R_{s,t}(G_1(x)) + R_{s,t}(G_2(x))$.
4. If G consists of subgraphs G_1 and G_2 connected in parallel as described above, then $(R_{s,t}(G(x)))^{-1} = (R_{s,t}(G_1(x)))^{-1} + (R_{s,t}(G_2(x)))^{-1}$.
5. If $G(x)$ is a subgraph of $G(y)$, then $R_{s,t}(G(x)) \geq R_{s,t}(G(y))$.
6. If s and t are connected in $G(x)$, then $R_{s,t}(G(x)) \leq d$, where d is the length of the shortest path from s to t.

Effective Capacitance

1. If G consists of a single edge between s and t, and $G(x)$ does not include the edge $\{s,t\}$, then $C_{s,t}(G(x)) = 1$.
2. If s and t are connected in $G(x)$, then $C_{s,t}(G(x) = \infty$.
3. If G consists of subgraphs G_1 and G_2 connected in series as described above, then $(R_{s,t}(G(x)))^{-1} = (R_{s,t}(G_1(x)))^{-1} + (R_{s,t}(G_2(x)))^{-1}$.
4. If G consists of subgraphs G_1 and G_2 connected in parallel, then $R_{s,t}(G(x)) = R_{s,t}(G_1(x)) + R_{s,t}(G_2(x))$.
5. If $G(x)$ is a subgraph of $G(y)$, then $C_{s,t}(G(x)) \leq C_{s,t}(G(y))$.
6. If s and t are not connected in $G(x)$, $C_{s,t}(G(x))$ is less than the size of the smallest cut in G between s and t.

We analyze our algorithms using these properties rather than first principles to demonstrate the relative ease of bounding the query complexity of span program algorithms for st-connectivity problems.

Now we can describe the performance of the span program algorithm for deciding st-connectivity:

▶ **Theorem 4.** *[10] Let $G = (V, E)$ be a graph with $s, t \in V(G)$. Then there is a span program algorithm whose bounded-error quantum query complexity of evaluating whether s and t are connected in $G(x)$ promised $x \in X$ and $X \subseteq \{0,1\}^N$ is*

$$O\left(\sqrt{\max_{\substack{x\in X \\ R_{s,t}(G(x))\neq\infty}} R_{s,t}(G(x)) \times \max_{\substack{x\in X \\ C_{s,t}(G(x))\neq\infty}} C_{s,t}(G(x))}\right). \tag{1}$$

Furthermore, the space complexity is

$$O(\max\{\log(|E|), \log(|V|)\}), \tag{2}$$

and the time complexity is U *times the query complexity, where* U *is the time required to perform one step of a quantum walk on G (see [11]).*

Ito and Jeffery describe an algorithm that can be used to estimate $R_{s,t}(G(x))$. It depends on a quantity called the negative witness size, which we denote $\widetilde{R}_-(x, G)$ (this is the quantity $\widetilde{w}_-(x)$ of [9] tailored to the case of the st-connectivity span program). Let $\mathcal{L}(U, \mathbb{R})$ be the set of linear maps from a set U to $\mathbb{R}$. Then we have the following definition:

▶ **Definition 5** (See Theorem 4.2, [9]). *Let $G = (V, E)$ with $s, t \in V$. If $G(x)$ is connected from s to t, let $V_x \subseteq V$ be the set of vertices connected to both s and t. Then there is a unique map $\mathcal{V}_x \in \mathcal{L}(V_x, \mathbb{R})$ such that $\mathcal{V}_x(s) = 1$, $\mathcal{V}_x(t) = 0$, and $\sum_{\{u,v\}\in E(G(x))} (\mathcal{V}_x(u) - \mathcal{V}_x(v))^2$ is minimized. Then the negative approximate witness size of input x on the graph G is*

$$\widetilde{R}_-(x, G) = \min_{\mathcal{V}\in\mathcal{L}(V,\mathbb{R}):\mathcal{V}(u)=\mathcal{V}_x(u) \text{ if } u\in V_x} \sum_{\{u,v\}\in E} (\mathcal{V}(u) - \mathcal{V}(v))^2 . \tag{3}$$

▶ **Theorem 6.** *Let G be a graph with $s, t \in V(G)$. Then the bounded-error quantum query complexity of estimating $R_{s,t}(G(x))$ to multiplicative error ϵ promised s and t are connected in $G(x)$ and $x \in X$ for $X \subseteq \{0,1\}^N$ is $\widetilde{O}\left(\epsilon^{-3/2}\sqrt{R_{s,t}(G(x))\widetilde{R}_-}\right)$, where $\widetilde{R}_- = \max_{x\in X} \widetilde{R}_-(x, G)$.*

3 Quantum Algorithms for Detecting and Characterizing Cycles

In this section, we prove the following results on detecting and characterizing cycles:

▶ **Theorem 7.** *Let G be the complete graph on n vertices. If we are promised that either $G(x)$ is connected with circuit rank at least r, or $G(x)$ is not connected and contains at most μ edges, then the bounded-error quantum query complexity of detecting a cycle in $G(x)$ is $O(\mu\sqrt{n/r})$.*

▶ **Theorem 8.** *Given a generic graph G with m edges, and a parameter $\epsilon \ll 1$ (here ϵ can be a constant or can depend on the input) there is a quantum algorithm that estimates the circuit rank r of $G(x)$ to multiplicative error ϵ using $\widetilde{O}\left(\epsilon^{-3/2}\sqrt{m\mu/r}\right)$ applications of O_x, under the promise that $G(x)$ has at most μ edges.*

A few notes on these theorems:

- Theorem 7 has a worst case upper bound of $O(n^{3/2})$, which matches the optimal lower bound. This is because $r \geq 1$ (r takes value 1 in the case of a single cycle) and $\mu \leq n - 1$ (since a graph without cycles must be a forest).
- In Theorem 8, if r and ϵ are $O(1)$ and if nothing is known about μ (in which case it could be as large as m), one would need to query all edges of the graph. However, given a promise that r is large, for example if $r = \Omega(m^\beta)$ for a positive constant β, or a promise on μ, we can do better than the trivial classical algorithm of querying all edges.

We prove both of these results using a reduction from cycle detection to *st*-connectivity. Specifically we construct a graph G_{cyc} such that $G_{\text{cyc}}(x)$ has an *st*-path if and only if $G(x)$ has a cycle. We note that there is a cycle in $G(x)$ if and only if an edge $\{u, v\}$ is present in $G(x)$ and there is a path from u to v in $G(x)$ that does not use the edge $\{u, v\}$. Thus, our reduction tests every edge in G to determine whether these two conditions are satisfied. We use the encoding of logical AND and OR into *st*-connectivity using serial and parallel composition, as described in Section 2 and in Refs. [14, 11].

We now describe how to build up G_{cyc} from simpler graphs. For an edge $\ell = \{u, v\} \in E(G)$ let G_ℓ^- be the graph that is the same as G, except with the edge ℓ removed, and the vertex u labeled as s, and the vertex v labeled as t. (The choice of which endpoint of ℓ is s and which is t is arbitrary - either choice is acceptable). Each edge in G_ℓ^- is associated with the same literal of x as the corresponding edge in G. Thus there is an *st*-path in $G_\ell^-(x)$ if and only if there is a path between u and v in $G(x)$ that does not go through $\{u, v\}$.

Next, for an edge $\ell \in E(G)$ let G^1_ℓ be the graph with exactly two vertices labeled s and t, and one edge between them. The one edge in G^1_ℓ is associated with the same literal bit of x as ℓ. Thus there is an st-path in $G^1_\ell(x)$ if and only if $\ell \in E(G(x))$.

Next, we create the graph G_ℓ by connecting G^1_ℓ and G^-_ℓ in series, while leaving the associations between edges and literals the same. Then because connecting st-connectivity graphs in series is equivalent to logical AND, there is an st-path through $G_\ell(x)$ if and only if there is a cycle in $G(x)$ passing through ℓ.

Finally, we create the graph G_{cyc} by connecting all of the graphs G_ℓ (for each $\ell \in E(G)$) in parallel, again retaining the association between edges and literals. Since attaching graphs in parallel is equivalent to logical OR, $G_{\text{cyc}}(x)$ has an st-path if and only if there is a cycle through some edge of $G(x)$. See Figure 1 for an example of the construction of G_{cyc}.

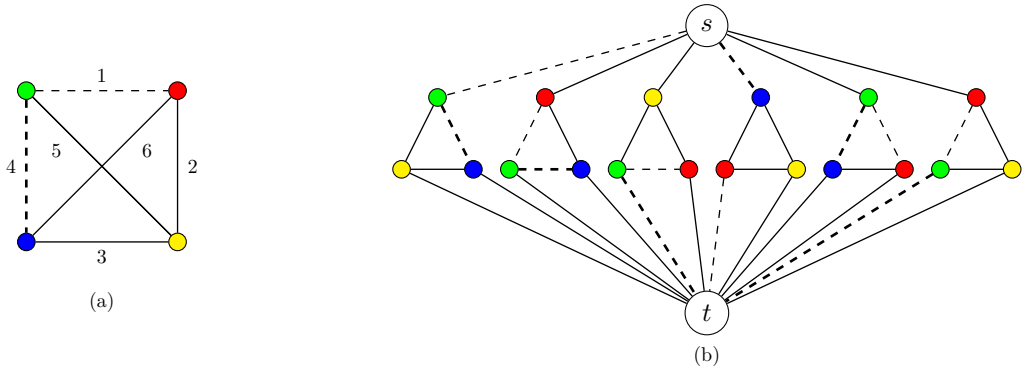


Figure 1 (a) A graph $G(x)$, where edges 2, 3, 5, and 6 are present (solid lines indicate the presence of an edge, dashed lines indicate the absence of an edge). (b) The graph $G_{\text{cyc}}(x)$ that $G(x)$ produces. There is a cycle involving the edges 2, 3, and 6 in $G(x)$, and thus there are paths from s to t in the subgraphs G_2, G_3, and G_6 in $G_{\text{cyc}}(x)$.

In order to use Theorem 4 to determine the query complexity of deciding st-connectivity on $G_{\text{cyc}}(x)$, we next analyze the effective resistance (respectively capacitance) of $G_{\text{cyc}}(x)$ in the presence (resp. absence) of cycles in $G(x)$. We first show the following relationship between effective resistance and circuit rank:

▶ **Lemma 9.** *Let r be the circuit rank of $G(x)$. Then*

$$R_{s,t}(G_{cyc}(x)) = \frac{1}{r}. \tag{4}$$

The proof of Lemma 9 uses the following result from Ref. [5] relating effective resistance and spanning trees:

▶ **Theorem 10** ([5])**.** *Let $\{u, v\} = \ell$ be an edge in a connected graph G. Then the effective resistance between vertices u and v is equal to the number of spanning trees that include an edge ℓ, divided by the total number of spanning trees:*

$$R_{u,v}(G) = \frac{t_\ell(G)}{t(G)}. \tag{5}$$

Proof of Lemma 9. Using the rules that the effective resistance of graphs in series adds, (Effective Resistance Property 3), and the inverse effective resistance of graphs in parallel

adds, (Effective Resistance Property 4), we have:

$$R_{s,t}(G_{\text{cyc}}(x)) = \left(\sum_{(u,v)\in E(G(x))} 1 - R_{u,v}(G(x)) \right)^{-1}. \tag{6}$$

(We include this relatively straightforward calculation in Appendix A.)

We next relate the righthand side of Equation (6) to the circuit rank. Let $G(x)$ be a graph with κ connected components. Let $g_i(x)$ be a subgraph consisting of the i^{th} connected component of G, with n_i vertices. We count the number of times edges are used in all spanning trees of $g_i(x)$ in two ways. First, we multiply the number of spanning trees by the number of edges in each spanning tree. Second, for each edge we add the number of spanning trees that include that edge. Setting these two terms equal, we have,

$$t(g_i(x))(n_i - 1) = \sum_{\ell\in E(g_i(x))} t_\ell(g_i(x)). \tag{7}$$

Rearranging, and using Theorem 10 we have

$$n_i - 1 = \sum_{\ell\in E(g_i(x))} \frac{t_\ell(g_i(x))}{t(g_i(x))} = \sum_{\{u,v\}\in E(g_i(x))} R_{u,v}(g_i(x)), \tag{8}$$

where if the sum has no terms (i.e. $E(g_i(x)) = \varnothing$), we define it to be zero.

Summing over all κ components of $G(x)$, we have

$$n - \kappa = \sum_{\{u,v\}\in E(G)} R_{u,v}(G(x)), \tag{9}$$

Finally, using the fact that

$$\sum_{\{u,v\}\in E(G(x))} 1 = m, \tag{10}$$

where m is the number of edges in $G(x)$, and combining with Equation (9), and Definition 1, we have

$$\sum_{\{u,v\}\in E(G(x))} 1 - R_{u,v}(G(x)) = r. \tag{11}$$

Finally Equation (11) and Equation (6) give the result. ◀

We next analyze the effective capacitance of $G_{\text{cyc}}(x)$ in the case of no cycles in $G(x)$:

▶ **Lemma 11.** *If G is the complete graph on n vertices and $G(x)$ has no cycles and at most μ edges, then, $C_{s,t}(G_{cyc}(x)) = O(n\mu^2)$.*

Proof. We first analyze the effective capacitance of the subgraph $G_\ell(x)$ in two cases, when $\ell \in E(G(x))$ and when $\ell \notin E(G(x))$.

When $\ell \in E(G(x))$, then using Effective Capacitance Properties 2 and 3, we have $C_{s,t}(G_\ell(x)) = C_{s,t}(G_\ell^-(x))$. Then using Effective Capacitance Property 6, we have that $C_{s,t}(G_\ell^-(x))$ is less than the size of the cut between vertices s and t in $G_\ell^-(x)$. Since there are μ edges and n vertices in $G(x)$, this quantity is bounded by $O(n\mu)$. (The worst case is when there are $\Omega(\mu)$ vertices connected to s, e.g.)

When $\ell \notin E(G(x))$, then using Effective Capacitance Properties 1 and 3, $C_{s,t}(G_\ell(x)) = O(1)$.

Since there are $n-\mu$ graphs $G_\ell(x)$ with $\ell \notin E(G(x))$ and μ graphs $G_\ell(x)$ with $\ell \in E(G(x))$, using Effective Capacitance Property 4 for graphs connected in parallel, we have that $C_{s,t}(G_{\text{cyc}}(x)) = O(\mu^2 n)$. ◀

In order to prove Theorem 8, we need to analyze $\widetilde{R}_-(x, G_{\text{cyc}})$:

▶ **Lemma 12.** *For a graph G with m edges, let $X = \{x : G(x) \text{contains a cycle and } |E(G(x))| \leq \mu\}$ and let $\widetilde{R}_- = \max_{x \in X} \widetilde{R}_-(x, G_{cyc})$. Then $\widetilde{R}_- = O(m\mu)$.*

Proof. Looking at Equation (3), for $\ell \notin E(G(x))$, we have that all $v \in V(G_\ell)$ (except s and t) are not in V_x, so any choice of $\mathcal{V}$ on these vertices will give an upper bound on the minimizing map. We choose $\mathcal{V}(v) = 0$ for these vertices to give us our bound, which contributes 1 to the sum for each such subgraph. Thus edges in these subgraphs contribute $m - \mu$ to the total.

For $\ell \in E(G(x))$, for vertices in these subgraphs which are also part of V_x, they will get mapped by $\mathcal{V}_x$ to values between 0 and 1 inclusive (since $\mathcal{V}_x$ can be seen as the voltage induced at each point by a unit potential difference between s and t). If we choose the remaining vertices to also get mapped to values between 0 and 1 by $\mathcal{V}$, we will again have an upper bound on the minimum. Then $(\mathcal{V}(u) - \mathcal{V}(v))^2 \leq 1$ across all edges in these subgraphs. Since there are m edges in each subgraph, and μ such subgraphs, edges in these subgraphs contributes $m\mu$ to the total.

Combining the two terms, we have that $\widetilde{R}_-(x, G_{\text{cyc}}) \leq m - \mu + m\mu = O(m\mu)$. ◀

Now we can put these results together to prove Theorem 8:

Proof of Theorem 8. Using Theorem 6, Lemma 9, and Lemma 12, we can estimate one over the circuit rank (i.e. $1/r$) to multiplicative error ϵ. That is, we get an estimate of $1/r$ within $(1 \pm \epsilon)/r$. Now if we take the inverse of this estimate, we get an estimate of of r within $r/(1 \pm \epsilon)$. But since $\epsilon \ll 1$, taking the Taylor expansion, we have $1/(1 \pm \epsilon) \approx (1 \pm \epsilon)$ to first order in ϵ. ◀

4 Algorithms for Detecting Odd Paths, Bipartiteness, and Even Cycles

In this section, we note that a slight variation on one of the st-connectivity problems considered by Cade et al. in Ref. [7] can be used to detect odd paths and bipartiteness; furthermore the bipartiteness testing algorithm we describe is optimal in query complexity, and far simpler than the bipartiteness algorithm in [7]. We then use a similar construction to create a reduction from even-length cycle detection to *st*-connectivity.

All of these algorithms involve the bipartite double graph of the original graph. Given a graph G with vertices u and v, let $K^G_{u,v}$ be the bipartite double graph of G, with vertex u_0 relabeled as s, and vertex v_1 relabeled as t. To define $K^G_{u,v}(x)$, if $\{x, y\} \in E(G)$ is associated with a literal, then $\{x_0, y_1\}$ and $\{x_1, y_0\}$ in $E(K^G_{u,v})$ are associated with the same literal.

We first show a reduction from detecting an odd-length path to *st*-connectivity on K^G:

▶ **Lemma 13.** *Let G be a graph with vertices u and v. There is an odd-length path from u to v in $G(x)$ if and only if there is an st-path in $K^G_{u,v}(x)$*

Proof. Suppose there is an odd-length path from u to v in $G(x)$. Let the path be $u, \eta^1, \eta^2, \ldots, \eta^k, v$ where k is an even integer greater than or equal to 0. Then there is a path $s, \eta^1_1, \eta^2_0, \ldots, \eta^k_0, t$, in $K^G_{u,v}(x)$ (where the path goes through $\eta^i_{i \bmod 2}$). For the other direction, if there is a path from s to t in $K^G_{u,v}(x)$, there is an odd-length path from u to v in G. Note that any path in $K^G_{u,v}(x)$ must alternate between 0- and 1-labeled vertices. If there is a path that starts at a 0-labeled vertex and ends at a 1-labeled vertex, it must be an odd-length path. Then there must be the equivalent path in $G(x)$, but without the labeling. See Figure 2 for an example of this reduction. ◀

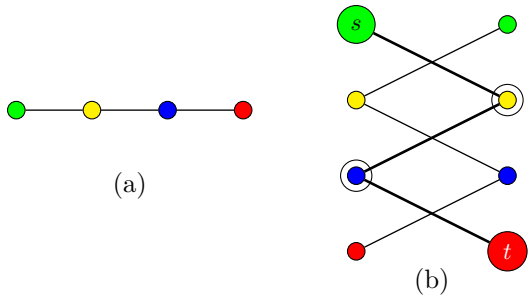


Figure 2 (a) A simple example of a graph G with an odd-length path between green and red vertices. (b) The bipartite double graph $K^G_{green,red}$ with a path between s and t.

▶ **Theorem 14.** *Let G be a graph with n vertices and m edges, with vertices u and v. Then there is a bounded-error quantum query algorithm that detects an odd length path from u to v in G using $O(\sqrt{nm})$ queries.*

Proof. Using Lemma 13 we reduce the problem to st-connectivity on $K^G_{u,v}$. Then using Theorem 4, we need to bound the largest effective resistance and effective capacitance of $K^G_{u,v}(x)$ for any string x. The longest possible path from s to t in $K^G_{u,v}(x)$ is $O(n)$ so by Effective Resistance Property 6, $R_{s,t}(K^G_{u,v}(x)) = O(n)$. The longest possible cut between s and t is $O(m)$, so by Effective Capacitance Property 6, $C_{s,t}(K^G_{s,t}(x)) = O(m)$. This gives the claimed query complexity. ◀

Note u_0 is connected to u_1 in $K^G(x)$ if and only if there is an odd-length path from u to itself in $G(x)$, where this path is allowed to double back on itself, as in Figure 3. This odd-length path in turn occurs if and only if the connected component of $G(x)$ that includes u is not bipartite (has an odd cycle)! Thus if we are promised that $G(x)$ is connected, we can pick any vertex in G, run the algorithm of Theorem 14 on $K^G_{u,u}(x)$, and determine if the graph is bipartite, which requires $O(\sqrt{nm})$ queries.

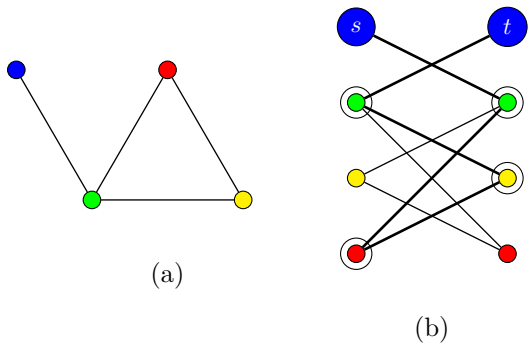


Figure 3 (a) A graph G with an odd cycle. (b) The bipartite double graph K^G with a path between the two green vertices.

On the other hand, if we are not promised that $G(x)$ is connected, we simply need to check whether there is an odd path from any of the n vertices of G to itself, and we now show that doing this check does not increase the query complexity. We use a similar strategy as with cycle detection:

▶ **Theorem 15.** *Let G be a graph with n vertices and m edges. Then there is a bounded-error quantum query algorithm that detects an odd cycle (in effect, non-bipartiteness) in $O(\sqrt{nm})$ queries.*

Proof Sketch. Let G_{bip} be the graph that consists of the the graphs $K^G_{u,u}$ composed in parallel for all $u \in V(G)$. This amounts to evaluating the logical OR of there being an odd cycle connected to any vertex in G.

A similar analysis as in cycle detection shows that the effective resistance of G_{bip} is $O(1)$, if there is an odd cycle. On the other hand, since there are n copies of K^G in this new graph, and each copy has m edges, the largest possible cut is $O(nm)$. Applying Theorem 4 gives the result. ◀

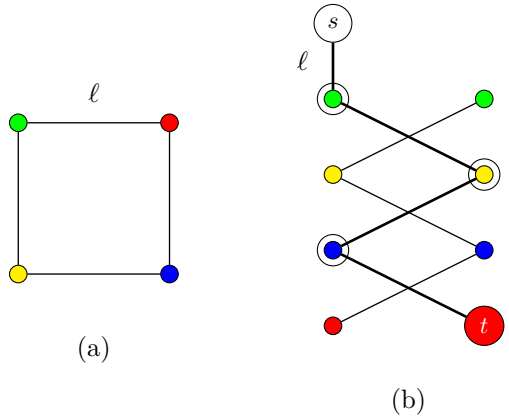


Figure 4 (a) An simple example of a graph G with an even-length cycle. (b) The bipartite double graph $K^G_{green,red}$ connected in series with $G^1_{\{red,green\}}$. We see there is a path from s to t in this graph, corresponding to an even-length cycle passing through ℓ.

Finally, we show how to detect even cycles:

▶ **Theorem 16.** *Let G be a graph with n vertices and m edges. Then there is a bounded-error quantum query algorithm that detects an even-length cycle in $O(\sqrt{nm})$ queries.*

Proof. For an edge $\ell = \{u, v\} \in E(G)$, note that there is an st-path in $K^{G^-_\ell}_{u,v}$ if and only if there is an odd-length path from u to v that does not use the edge ℓ itself. Thus if we consider the graph composed of G^1_ℓ and $K^{G^-_\ell}_{u,v}$ in series, which we denote G^E_ℓ, there is an st-path if and only if there is an even-length cycle through ℓ. Finally, if we compose the graphs G^E_ℓ in parallel for all $\ell \in E(G)$, we obtain a graph that has an st-path if and only if there is an even cycle passing through some edge in G, as in Figure 4.

As in our previous analyses of cycle detection and bipartiteness, if there is an even cycle, the effective resistance will be $O(1)$. On the other hand, if there is no even-length cycle, then it is a fairly well known fact that the number of edges in G is $O(n)$. Then similar to previous analyses, for each graph G^E_ℓ such that $\ell \in E(G)$, we have that the cut is $O(m)$. Otherwise, for each graph G^E_ℓ such that $\ell \in E(G)$, we have that the cut is $O(1)$. Thus a bound on the size of the total cut is $O(n^2 + nm) = O(nm)$ (assuming that $n = O(m)$.) Applying Theorem 4 gives the result. ◀

References

1 C. Alvarez and R. Greenlaw. A compendium of problems complete for symmetric logarithmic space. *Computational Complexity*, 9(2):123–145, 2000.

2 A. Āriņš. Span-program-based quantum algorithms for graph bipartiteness and connectivity. In *International Doctoral Workshop on Mathematical and Engineering Methods in Computer Science*, pages 35–41. Springer, 2015.

3 A. Belovs. Span programs for functions with constant-sized 1-certificates. In *Proceedings of the 44th Symposium on Theory of Computing (STOC 2012)*, pages 77–84, 2012.

4 A. Belovs and B. W. Reichardt. Span programs and quantum algorithms for *st*-connectivity and claw detection. In *Proceedings of the 20th European Symposium on Algorithms (ESA 2012)*, pages 193–204, 2012.

5 N. Biggs. Algebraic Potential Theory on Graphs. *Bulletin of the London Mathematical Society*, 29, 1997.

6 B. Bollobás. *Modern graph theory*. Springer Science & Business Media, 2013.

7 C. Cade, A. Montanaro, and A. Belovs. TIME AND SPACE EFFICIENT QUANTUM ALGORITHMS FOR DETECTING CYCLES AND TESTING BIPARTITENESS. *Quantum Information and Computation*, 18(1&2):0018–0050, 2018.

8 A. M. Childs and R. Kothari. Quantum query complexity of minor-closed graph properties. *SIAM Journal on Computing*, 41(6):1426–1450, 2012.

9 T. Ito and S. Jeffery. Approximate span programs. *Algorithmica*, pages 1–38, 2015.

10 M. Jarret, S. Jeffery, S. Kimmel, and A. Piedrafita. Quantum Algorithms for Connectivity and Related Problems. In *26th Annual European Symposium on Algorithms (ESA 2018)*, pages 49:1–49:13, 2018. doi:10.4230/LIPIcs.ESA.2018.49.

11 S. Jeffery and S. Kimmel. Quantum algorithms for graph connectivity and formula evaluation. *Quantum*, 1:26, August 2017. doi:10.22331/q-2017-08-17-26.

12 M. Karchmer and A. Wigderson. On span programs. In *Proceedings of the 8th Annual IEEE Conference on Structure in Complexity Theory*, pages 102–111, 1993.

13 T. J. McCabe. A complexity measure. *IEEE Transactions on software Engineering*, SE-2(4):308–320, 1976.

14 N. Nisan and A. Ta-Shma. Symmetric Logspace is Closed Under Complement. In *Proceedings of the Twenty-seventh Annual ACM Symposium on Theory of Computing (STOC 1995)*, pages 140–146. ACM, 1995. doi:10.1145/225058.225101.

15 B. W. Reichardt. Span programs and quantum query complexity: The general adversary bound is nearly tight for every boolean function. In *Foundations of Computer Science, 2009. FOCS'09. 50th Annual IEEE Symposium on*, pages 544–551. IEEE, 2009.

16 B. W. Reichardt. Reflections for quantum query algorithms. In *Proceedings of the 22nd Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2011)*, pages 560–569, 2011.

17 R. Yuster and U. Zwick. Finding even cycles even faster. *SIAM Journal on Discrete Mathematics*, 10(2):209–222, 1997.

18 A. Zamora. An Algorithm for Finding the Smallest Set of Smallest Rings. *Journal of Chemical Information and Computer Sciences*, 16(1):40–43, 1976. doi:10.1021/ci60005a013.

19 S. Zhang. On the power of Ambainis lower bounds. *Theoretical Computer Science*, 339(2-3):241–256, 2005.

A Effective Resistance of G_{cyc}

We relate the effective resistance across edges in G to the effective resistance between s and t in G_{cyc}. (The proof also applies to $G(x)$ and $G_{\text{cyc}}(x)$.) We will write $R_{s,t}(G_{\text{cyc}})$ in terms of a sum of $R_{u,v}(G)$ where (u, v, ℓ) is an edge on a cycle in G.

Consider an edge $\{u, v\} \in E(G)$. Then using Effective Resistance Property 4 (for graphs composed in parallel), we have

$$\frac{1}{R_{u,v}(G)} = 1 + \frac{1}{R_{s,t}(G_\ell^-)}. \tag{12}$$

Rearranging, we have

$$R_{s,t}(G_\ell^-) = \frac{R_{u,v}(G)}{1 - R_{u,v}(G)} \tag{13}$$

Then using Effective Resistance Property 3 (for graphs composed in series), we have that

$$\begin{aligned} R_{s,t}(G_\ell) &= R_{s,t}(G_\ell^-) + 1 \\ &= \frac{R_{u,v}(G)}{1 - R_{u,v}(G)} + 1 \\ &= \frac{1}{1 - R_{u,v}(G)}. \end{aligned} \tag{14}$$

Finally, using Effective Resistance Property 4 (graphs composed in parallel) again, we have

$$\frac{1}{R_{s,t}(G_{\text{cyc}})} = \sum_{\{u,v,\}\in E(G)} 1 - R_{u,v}(G). \tag{15}$$

Bayesian ACRONYM Tuning

John Gamble
Quantum Architectures and Computing Group, Microsoft Research, Redmond WA, USA
john.gamble@microsoft.com

Christopher Granade
Quantum Architectures and Computing Group, Microsoft Research, Redmond WA, USA
christopher.granade@microsoft.com

Nathan Wiebe
Quantum Architectures and Computing Group, Microsoft Research, Redmond WA, USA
nathanwiebe@gmail.com

Abstract

We provide an algorithm that uses Bayesian randomized benchmarking in concert with a local optimizer, such as SPSA, to find a set of controls that optimizes that average gate fidelity. We call this method Bayesian ACRONYM tuning as a reference to the analogous ACRONYM tuning algorithm. Bayesian ACRONYM distinguishes itself in its ability to retain prior information from experiments that use nearby control parameters; whereas traditional ACRONYM tuning does not use such information and can require many more measurements as a result. We prove that such information reuse is possible under the relatively weak assumption that the true model parameters are Lipschitz-continuous functions of the control parameters. We also perform numerical experiments that demonstrate that over-rotation errors in single qubit gates can be automatically tuned from 88% to 99.95% average gate fidelity using less than $1kB$ of data and fewer than 20 steps of the optimizer.

2012 ACM Subject Classification Hardware → Quantum computation

Keywords and phrases Quantum Computing, Randomized Benchmarking

Digital Object Identifier 10.4230/LIPIcs.TQC.2019.7

Acknowledgements This project was prepared using a reproducible workflow [16].

1 Introduction

Tuning gates in quantum computers is a task of fundamental importance to building a quantum computer. Without tuning, most quantum computers would have insufficient accuracy to implement a simple algorithm let alone achieve the stringent requirements on gate fidelity imposed by quantum error correction [12, 4]. Historically, qubit tuning has largely been done by experimentalists refining an intelligent initial guess for the physical parameters by hand to account for the ideosyncracies of the device. Recently, alternatives have been invented that allow devices to be tuned in order to improve performance on real-world estimates of gate quality. These methods, often based on optimizing quantities such as average gate fidelities, are powerful but come with two drawbacks. At present all such methods require substantial input data to compute the average gate fidelity and estimate its gradient, and at present no method can use information from the history of an optimization procedure to reduce such data needs. Our approach, which we call Bayesian ACRONYM tuning (or BACRONYM), addresses these problems.

BACRONYM is based strongly on the ACRONYM protocol invented by Ferrie and Moussa [11]. There are two parts to the ACRONYM gate tuning protocol. The first uses randomized benchmarking [25] to obtain an estimate of gate fidelity as a function of the controls. The second optimizes the average gate fidelity using a local optimizer such as Nelder-Mead or stochastic gradient descent. While many methods can be used to estimate the average

gate fidelity, randomized benchmarking is of particular significance because of its ability to give an efficient estimate of the average gate fidelity under reasonable assumptions [30], and because of its amenability to experimental application [17]. The algorithm then uses a protocol, similar to SPSA (Simultaneous Perturbation Stochastic Approximation) [33], to optimize the estimate of the gate fidelity by changing the experimental controls and continues to update the parameters until the desired tolerance is reached.

The optimization used in ACRONYM simply involves varying a parameter slightly and applying the fidelity estimation protocol from scratch every time. When the a quantum system is evaluated at two nearby points in parameter space, an operation performed repeatedly in descent algorithms, the objective function does not typically change much in practice. Since ACRONYM does not take this into account, it requires more data than is strictly needed. Thus, if ACRONYM could be modified to use prior information extracted from the previous iteration in SPSA, the data needed to obtain an estimate of the gradient can be reduced.

Bayesian methods provide a natural means to use prior information within parameter estimation and have been used previously to analyze randomized benchmarking experiments. These methods, yield estimates of the average gate fidelity based on prior beliefs of about the randomized benchmarking parameters as well as the evidence obtained experimentally [13]. To use a Bayesian approach, we begin by taking as input a probability distribution for the average gate fidelity (AGF) as function of the control parameters $\boldsymbol{\theta}$, $\Pr(\text{AGF}|\boldsymbol{\theta})$. This is our prior belief about the average gate fidelity. In addition to a prior, we need a method for computing the likelihood of witnessing a set of experimental evidence E. This is known as the likelihood function; in the case of Bayesian randomized benchmarking, it is $\Pr(E|\text{AGF};\boldsymbol{\theta})$. Given these as input, we then seek to output an approximation to the posterior probability distribution, *i.e.,* the probability with which the AGF takes a specific value conditioned on our prior belief and E. To accomplish this, we use Bayes' theorem, which states that

$$\Pr(\text{AGF}|E;\boldsymbol{\theta}) = \frac{\Pr(\text{AGF}|\theta)\Pr(E|\text{AGF};\theta)}{\Pr(E|\theta)}, \tag{1}$$

where $\Pr(E|\theta)$ is just a normalization constant. From the posterior distribution $\Pr(\text{AGF}|E;\boldsymbol{\theta})$ we can then extract a point estimate of the AGF (by taking the mean) or estimate its uncertainty (by computing the variance).

Our work combines these two ideas to show that provided the quantum channels that describe the underlying gates are continuous functions of the control parameters then the uncertainty in parameters like AGF that occurs from transitioning from $\boldsymbol{\theta} \to \boldsymbol{\theta'}$ in the optimization process is also a continuous function of $\|\boldsymbol{\theta} - \boldsymbol{\theta'}\|$. This gives us a rule that we can follow to argue how much uncertainty we have to add to our posterior distribution $\Pr(\text{AGF}|E;\boldsymbol{\theta})$ to use it as a prior $\Pr(\text{AGF}|\boldsymbol{\theta'})$ at the next step of the gradient optimization procedure.

1.1 Notation

The notation that we use in this paper necessarily spans several fields, most notably Bayesian inference and randomized benchmarking theory. Here we will introduce the necessary notation from these fields in order to understand our results. For any distribution $\Pr(\boldsymbol{x})$ over a vector $\boldsymbol{x}$ of random variables, we write $\text{supp}(\Pr(\boldsymbol{x}))$ to mean the set of vectors $\boldsymbol{x}$ such that $\Pr(\boldsymbol{x}) > 0$. When it is clear from context, we will write $\text{supp}(\boldsymbol{x}|\boldsymbol{y})$ in place of $\text{supp}(\Pr(\boldsymbol{x}|\boldsymbol{y}))$.

Let $\mathcal{H} = \mathbb{C}^d$ be a finite-dimensional Hilbert space describing the states of a quantum system of interest, and let $\text{L}(\mathcal{H})$ be the set of linear operators acting on $\mathcal{H}$. Let $\text{Herm}(\mathcal{H}) \subsetneq \text{L}(\mathcal{H})$ and $\text{U}(\mathcal{H}) \subsetneq \text{L}(\mathcal{H})$ be the sets of Hermitian and unitary operators acting on $\mathcal{H}$, respectively. For the

most part, however, we are not concerned directly with pure states $|\psi\rangle \in \mathcal{H}$, but with classical distributions over such states, described by density operators $\rho \in \mathrm{D}(\mathcal{H}) \subsetneq \mathrm{Herm}(\mathcal{H}) \subsetneq \mathrm{L}(\mathcal{H})$. Whereas $\mathcal{H}$ transforms under $\mathrm{U}(\mathcal{H})$ by left action, $\mathrm{D}(\mathcal{H})$ transforms under $\mathrm{U}(\mathcal{H})$ by the *group action* $\bullet : \mathrm{U}(\mathcal{H}) \times \mathrm{L}(\mathcal{H}) \to \mathrm{L}(\mathcal{H})$, given by $U \bullet \rho := U\rho U^\dagger$. We note that $\bullet$ is linear in its second argument, such that for a particular $U \in \mathrm{U}(\mathcal{H})$, $U\bullet : \mathrm{L}(\mathcal{H}) \to \mathrm{L}(\mathcal{H})$ is a linear function. We thus write that $U \bullet \in \mathrm{L}(\mathrm{L}(\mathcal{H}))$. Moreover, since $U \bullet$ is a completely positive and trace preserving map on $\mathrm{L}(\mathcal{H})$, we say that $U \bullet$ is a *channel* on $\mathcal{H}$, written $\mathrm{C}(\mathcal{H}) \subsetneq \mathrm{L}(\mathrm{L}(\mathcal{H})) \subsetneq \mathrm{L}(\mathcal{H}) \to \mathrm{L}(\mathcal{H})$. More generally, we take $\mathrm{C}(\mathcal{H})$ to be the set of all such completely positive and trace preserving maps acting on $\mathrm{L}(\mathcal{H})$.

1.2 Problem Description

Before proceeding further, it is helpful to carefully define the problem that we address with BACRONYM. In particular, let $G = \langle V_0, \ldots, V_{\ell-1}\rangle \subsetneq \mathrm{U}(\mathcal{H})$ be a group and a unitary 2-design [5], such that G is appropriate for use in standard randomized benchmarking. Often, G will be the Clifford group acting on a Hilbert space of dimension d, but smaller twirling groups may be chosen in some circumstances [18]. We will consider that the generator T is a gate, which we would like to tune to be V_0 without loss of generality, as a function of a vector $\boldsymbol{\theta}$ of control parameters, such that $T = T(\boldsymbol{\theta})$. We write that $V_i \perp\!\!\!\perp \boldsymbol{\theta}$ for all $i \geq 0$ to indicate that the generators $\{V_0, \ldots, V_{\ell-1}\}$ are not functions of the controls $\boldsymbol{\theta}$ (note that V_0 is manifestly not a function of the controls because it represents the ideal action). Nonetheless, it is often convenient to write that $V_i = V_i(\boldsymbol{\theta})$) with the understanding that $\partial_{\theta_j} V_i = 0$ for all $i \geq 0$ and for all control parameters θ_j.

In order to reason about the errors in our implementation of each generator, we will write that the imperfect implementation $\tilde{V} \in \mathrm{C}(\mathcal{H})$ of a generator $V \in \{V_0, \ldots, V_{\ell-1}\}$ is defined as

$$\tilde{V} = \Lambda_V(V \bullet) \tag{2}$$

which acts on ρ as

$$\tilde{V}[\rho] = \Lambda_V[V\rho V^\dagger], \tag{3}$$

where Λ_V is the *discrepancy channel* describing the errors in V. Note that for an ideal implementation, Λ_V is the identity channel.

We extend this definition to arbitrary elements of G in a straightforward fashion. Let $U := \prod_{i \in \boldsymbol{i}(U)} V_i$, where $\boldsymbol{i}(U)$ is the sequence of indices of each generator in the decomposition of U. For instance, if $G = \langle H, S\rangle$ for the phase gate $S = \mathrm{diag}(1, i)$, then $\sqrt{X} = HSH$ is represented by $\boldsymbol{i}(U) = (0, 1, 0)$. Combining the definition of U with Eq. (2), the imperfect composite action $\tilde{U}$ is

$$\tilde{U} = \prod_{i \in \boldsymbol{i}(U)} \tilde{V}_i = \prod_{i \in \boldsymbol{i}(U)} \Lambda_{V_i}(V_i \bullet) := \Lambda_U(U\bullet), \tag{4}$$

where the final point defines the composite discrepancy channel Λ_U. By rearranging the equation above, we obtain

$$\Lambda_U = \tilde{U}(U^\dagger \bullet) = \left(\prod_{i \in \boldsymbol{i}(U)} \Lambda_{V_i}(V_i \bullet)\right)(U^\dagger \bullet). \tag{5}$$

Returning to the example $\sqrt{X} = HSH$, we thus obtain that

$$\Lambda_{\sqrt{X}} = \Lambda_H(H\,\bullet\,)\Lambda_S(S\,\bullet\,)\Lambda_H(H\,\bullet\,)((H^\dagger S^\dagger H^\dagger)\,\bullet\,) \tag{6}$$

is the discrepancy channel describing the noise incurred if we implement $\widetilde{\sqrt{X}}$ as the sequence $\tilde{H}\tilde{S}\tilde{H}$.

Equipped with the discrepancy channels for all elements of G, we can now concretely state the parameters of interest to randomized benchmarking over G. Standard randomized benchmarking without sequence reuse [13], in the limit of long sequences [35], depends only on the state preparation and measurement (SPAM) procedure and on the average gate fidelity $\mathrm{AGF}(\Lambda_{\mathrm{ref}})$, where

$$\Lambda_{\mathrm{ref}} := \mathbb{E}_{U\sim\mathrm{Uni}(G)}[\Lambda_U] = \frac{1}{|G|}\sum_{U\in G}\Lambda_U \tag{7}$$

is the reference discrepancy channel, obtained by taking the expectation value of the discrepancy channel Λ_U over U sampled uniformly at random from G, and where the average gate fidelity is given by the expected action of a channel Λ over the Haar measure $\mathrm{d}\psi$,

$$\mathrm{AGF}(\Lambda) := \int \mathrm{d}\psi \langle\psi \mid \Lambda(|\psi\rangle\langle\psi|) \mid \psi|.\rangle \tag{8}$$

When discussing the quality of a particular generator, say $T := \tilde{V}_0$, we unfortunately cannot directly access $\mathrm{AGF}(\Lambda_T)$ experimentally. However, interleaved randomized benchmarking allows us to rigorously estimate $\mathrm{AGF}(\Lambda_T\Lambda_{\mathrm{ref}})$ in the limit of long sequences and without sequence reuse.

Our goal here is to find a set of control parameters that optimizes $\mathrm{AGF}(\Lambda_T\Lambda_{\mathrm{ref}})$. To state this more formally, suppose that T is a function of a vector $\boldsymbol{\theta}$ of control parameters such that $T = T(\boldsymbol{\theta})$. For all ideal generators, we write that $V_i \perp\!\!\!\perp \boldsymbol{\theta}$ for all $i \geq 0$ to indicate that the other generators $\{V_0, \ldots, V_{\ell-1}\}$ are not functions of the controls $\boldsymbol{\theta}$. We also assume that $\tilde{V}_i \perp\!\!\!\perp \boldsymbol{\theta}$ for all $i > 0$, so that $T(\boldsymbol{\theta}) = \Lambda_{V_0}(\boldsymbol{\theta})V_0$ is the sole generator we are optimizing. We therefore aim to find $\boldsymbol{\theta}$ such that $\boldsymbol{\theta} = \mathrm{argmax}\left(\mathrm{AGF}(\Lambda_{T(\boldsymbol{\theta})}\Lambda_{\mathrm{ref}})\right)$.

This problem has previously been considered by [7] and later by [22], who proposed the use of interleaved randomized benchmarking with least-squares fitting to implement an approximate oracle for $\mathrm{AGF}(\Lambda_T(\boldsymbol{\theta})\Lambda_{\mathrm{ref}}(\boldsymbol{\theta}))$. Taken together with the bounds showed by [26] and later improved by [23], this approximate oracle provides an approximate lower bound on $\mathrm{AGF}(\Lambda_{\mathrm{ref}}(\boldsymbol{\theta}))$. This lower bound can then be taken as an objective function for standard optimization routines such as Nelder–Mead to yield a "fix-up" procedure that improves gates based on experimental evidence. [11] showed an improvement in this procedure by the use of an optimization algorithm that is more robust to the approximations incurred by the use of finite data in the underlying randomized benchmarking experiments. In particular, the simultaneous pertubative stochastic approximation (SPSA) [33], while less efficient for optimizing exact oracles, can provide dramatic improvements in approximate cases such as that considered by [11]. This advantage has been further shown in other areas of quantum information, such as in tomography [10, 2].

We improve this result still further by using a Lipschitz continuity assumption on the dependence of Λ_T on $\boldsymbol{\theta}$ to propagate prior information between optimization iterations. This assumption is physically well-motivated: it reflects a desire that our control knobs have a smooth (but not known) influence on our generators. Since small gradient steps cannot greatly modify the average gate fidelity of interest under such a continuity assumption, the prior distribution for each randomized benchmarking experiment is closely related to the posterior distribution from the previous optimization iteration.

Recent work has shown, however, that this approach faces two significant challenges. First, the work of [30] has shown explicit counterexamples in which reconstructing $\mathrm{AGF}(\Lambda_T(\boldsymbol{\theta}))$ from $\mathrm{AGF}(\Lambda_T(\boldsymbol{\theta})\Lambda_{\mathrm{ref}}(\boldsymbol{\theta}))$ can yield very poor estimates due to the gauge dependence of this inverse problem. Second, the work of [19] has shown that the statistical inference problem induced by randomized benchmarking becomes considerably more complicated with sequence reuse, and in particular, depends on higher moments such as the unitarity [34]. While the work of [19] provides the first concrete algorithm that allows for learning randomized benchmarking parameters with sequence reuse, we will consider the single-shot limit to address the [30] argument, as this is the unique randomized benchmarking protocol that provides gauge invariant estimates of $\mathrm{AGF}(\Lambda_T(\boldsymbol{\theta})\Lambda_{\mathrm{ref}}(\boldsymbol{\theta}))$ [31], and as this model readily generalizes to include the effects of error correction [3].

In this work, we adopt as our objective function

$$F(\boldsymbol{\theta}) := \mathrm{AGF}(\Lambda_T(\boldsymbol{\theta})\Lambda_{\mathrm{ref}}(\boldsymbol{\theta})). \tag{9}$$

This choice of objective represents that we want to see improvements in the interleaved average gate fidelity, regardless of whether they occur from a more accurate target gate or a more accurate reference channel. In practice, these two contributions to our objective function can be teased apart by the use of more complete protocols such as gateset tomography [27, 1]. We proceed in three steps. First, we demonstrate that the Lipschitz continuity of $\Lambda_T(\boldsymbol{\theta})$ implies the Lipschitz continuity of $F(\boldsymbol{\theta})$. We then proceed to show that this implies an upper bound on $\mathrm{Var}[F(\boldsymbol{\theta}+\boldsymbol{\delta\theta})|\mathrm{data}]$ in terms of $\mathrm{Var}[F(\boldsymbol{\theta})|\mathrm{data}]$, such that we can readily produce estimates $\hat{F}(\boldsymbol{\theta})$ at each step of an optimization procedure, while reusing much of our data to accelerate the process. Finally, we conclude by presenting a numerical example for a representative model to demonstrate how BACRONYM may be used in practice.

2 Lipschitz Continuity of $F(\theta)$

Proving Lipshitz continuity of the objective function is an important first step towards arguing that we can reuse information during BACRONYM's optimization process. We need this fact because if the objective function were to vary unpredictably at adjacent values of the controls then finding the optima would reduce to an unstructured search problem, which cannot be solved efficiently. Our aim is to first argue that continuity of Λ implies continuity of F. We then will use this fact to argue about the maximum amount that the posterior variance can grow as the control parameters are updated, which will allow us to quantify how to propagate uncertainties of F at adjacent points later. We begin by recalling the definition of Lipschitz continuity for functions acting on vectors.

▶ **Definition 1** (Lipschitz continuity). *Given a Euclidean metric space S, a function $f : S \to \mathbb{R}$ is said to be Lipschitz continuous if there exists $\mathcal{L} \geq 0$ such that for all $\boldsymbol{x}, \boldsymbol{y} \in S$,*

$$|f(\boldsymbol{x}) - f(\boldsymbol{y})| \leq \mathcal{L}\|\boldsymbol{x} - \boldsymbol{y}\|. \tag{10}$$

If not otherwise stated, we will assume $\|\cdot\|$ on vectors to be the Euclidean norm $\|\cdot\|_2$.

As an example, $f(x) = \sqrt{x}$ is not Lipschitz continuous on [0,1], but any differentiable function on a closed, bounded interval of the real line is. We now generalize the notion of Lipschitz continuity to channels. Let $\mathrm{L}(\mathcal{H})$ be the set of all linear operators acting on the Hilbert space $\mathcal{H}$, and let $\mathrm{L}(\mathrm{L}(\mathcal{H}))$ be the set of linear operators acting on all such linear operators (often referred to as *superoperators*).

▶ **Definition 2** (Lipschitz continuity of channels)**.** *Given a metric space S and a Hilbert space $\mathcal{H}$, we say that a function $\Lambda : S \to \mathrm{L}(\mathrm{L}(\mathcal{H}))$ is $\mathcal{L}$-continuous or Lipschitz continuous in the $\star$ distance if there exists $\mathcal{L} \geq 0$ such that for all $\boldsymbol{x}, \boldsymbol{y} \in S$ and $\rho \in \mathrm{D}(\mathcal{H})$,*

$$\|\Lambda(\boldsymbol{x})[\rho] - \Lambda(\boldsymbol{y})[\rho]\|_\star \leq \mathcal{L}\|\boldsymbol{x} - \boldsymbol{y}\|. \tag{11}$$

If not specified explicitly, the trace norm $\|\cdot\| = \|\cdot\|_{Tr}$ is assumed for operators in $\mathrm{L}(\mathcal{H})$.

From the definition, we immediately can show the following:

▶ **Lemma 3** (Composition of Lipschitz continuous channels)**.** *Let $\Lambda, \Phi : S \to \mathrm{L}(\mathrm{L}(\mathcal{H}))$ be Lipschitz continuous in the trace distance with constants $\mathcal{L}$ and $\mathcal{M}$, respectively. Then, $(\Phi\Lambda) : \boldsymbol{x} \mapsto \Phi(\boldsymbol{x})\Lambda(\boldsymbol{x})$ is Lipschitz continuous in the trace distance with constant $\mathcal{L} + \mathcal{M}$.*

Proof. The proof of the lemma follows immediately after a few applications of the triangle inequality under the assumption of continuity of the individual channels.

$$\begin{aligned}
\|(\Phi\Lambda)(\boldsymbol{x})[\rho] - (\Phi\Lambda)(\boldsymbol{y})[\rho]\|_{\mathrm{Tr}} &= \|\Phi(\boldsymbol{x})[\Lambda(\boldsymbol{x})[\rho]] - \Phi(\boldsymbol{y})[\Lambda(\boldsymbol{y})\rho]]\|_{\mathrm{Tr}} \\
&= \|\Phi(\boldsymbol{x})[\Lambda(\boldsymbol{x})[\rho]] - \Phi(\boldsymbol{x})[\Lambda(\boldsymbol{y})[\rho]] \\
&\quad + \Phi(\boldsymbol{x})[\Lambda(\boldsymbol{y})[\rho]] - \Phi(\boldsymbol{y})[\Lambda(\boldsymbol{y})[\rho]]\|_{\mathrm{Tr}} \\
&\leq \|\Phi(\boldsymbol{x})[\Lambda(\boldsymbol{x})[\rho]] - \Phi(\boldsymbol{x})[\Lambda(\boldsymbol{y})[\rho]]\|_{\mathrm{Tr}} \\
&\quad + \|\Phi(\boldsymbol{x})[\Lambda(\boldsymbol{y})[\rho]] - \Phi(\boldsymbol{y})[\Lambda(\boldsymbol{y})[\rho]]\|_{\mathrm{Tr}} \\
&\leq \|\Phi(\boldsymbol{x})[\Lambda(\boldsymbol{x})[\rho]] - \Phi(\boldsymbol{x})[\Lambda(\boldsymbol{y})[\rho]]\|_{\mathrm{Tr}} + \mathcal{M}\|\boldsymbol{x} - \boldsymbol{y}\| \\
&\leq \|\Lambda(\boldsymbol{x})[\rho] - \Lambda(\boldsymbol{y})[\rho]\|_{\mathrm{Tr}} + \mathcal{M}\|\boldsymbol{x} - \boldsymbol{y}\| \\
&\leq \mathcal{L}\|\boldsymbol{x} - \boldsymbol{y}\| + \mathcal{M}\|\boldsymbol{x} - \boldsymbol{y}\|,
\end{aligned}$$

where the second-to-last line follows from contradiction on Helstrom's theorem [36]. ◀

We note that the above lemma immediately implies that if $\Lambda(\boldsymbol{\theta})$ is Lipschitz continuous in the trace distance with constant $\mathcal{L}$, then so is $(\Phi\Lambda)(\boldsymbol{\theta})$ for any channel $\Phi \perp\!\!\!\perp \boldsymbol{\theta}$, since Φ can be written as a channel that is Lipschitz continuous in the trace distance with constant 0.

▶ **Corollary 4** (Composition of multiple Lipschitz continuous functions and channels)**.** *Let $\Lambda_0, \Lambda_1, ..., \Lambda_k : S \to \mathrm{L}(\mathrm{L}(\mathcal{H}))$ be Lipschitz continuous in the trace distance with constants $\mathcal{L}_i$ with $i \in [0, 1, ..., k]$. Then, $(\Lambda_0\Lambda_1 \cdots \Lambda_k) : \boldsymbol{x} \mapsto \Lambda_0(\boldsymbol{x})\Lambda_1(\boldsymbol{x}) \cdots \Lambda_k(\boldsymbol{x})$ is Lipschitz continuous in the trace distance with constant $\sum_{i=0}^{k} \mathcal{L}_i$.*

▶ **Lemma 5.** *Let $\Lambda : S \to \mathrm{L}(\mathrm{L}(\mathcal{H}))$ be a convex combination of channels,*

$$\Lambda(\boldsymbol{\theta}) = \sum_i p_i \Lambda_i(\boldsymbol{\theta}), \tag{12}$$

where $\{p_i\}$ are nonnegative real numbers such that $\sum_i p_i = 1$, and where each $\Lambda_i : S \to \mathrm{L}(\mathrm{L}(\mathcal{H}))$ is Lipschitz continuous in a norm $\|\cdot\|_\star$ with constant $\mathcal{L}_i$. Then, Λ is Lipschitz continuous with constant $\bar{\mathcal{L}} = \sum_i p_i \mathcal{L}_i$.

Proof. Consider an input state $\rho \in \mathrm{D}(\mathcal{H})$. Then,

$$\begin{aligned}
\|\Lambda(\boldsymbol{\theta})[\rho] - \Lambda(\boldsymbol{\theta}')[\rho]\|_\star &= \left\| \sum_i p_i \left(\Lambda_i(\boldsymbol{\theta})[\rho] - \Lambda_i(\boldsymbol{\theta}')[\rho] \right) \right\|_\star \\
&\leq \sum_i p_i \left(\left\| \Lambda_i(\boldsymbol{\theta})[\rho] - \Lambda_i(\boldsymbol{\theta}')[\rho] \right\|_\star \right) \\
&\leq \sum_i p_i \mathcal{L}_i \|\boldsymbol{\theta} - \boldsymbol{\theta}'\| \\
&= \bar{\mathcal{L}} \|\boldsymbol{\theta} - \boldsymbol{\theta}'\|.
\end{aligned}$$

◀

Table 1 A partitioning of the twirling group $G = \langle H, S \rangle$ based on the number of occurrences of the target gate $T = S$ in the expansion of each element.

G_0	$\{\mathbb{1}, H\}$
G_1	$\{S, HS, SH, HSH\}$
G_2	$\{SS, HSS, SHS, SSH, HSHS, HSSH\}$
G_3	$\{SSS, HSSS, SHSS, SSHS, HSHSS, HSSHS, SHSSH, HSHSSH\}$
G_4	$\{SHSSS, SSHSS, HSHSSS, HSSHSS\}$

The above lemmas can then be used to show that $\mathrm{AGF}(\Lambda_T(\boldsymbol{\theta}))$ is Lipschitz continuous with constant $\mathcal{L}$ when $\Lambda_T(\boldsymbol{\theta})$ is Lipschitz continuous in the trace distance with constant $\mathcal{L}$, as we formally state in the following theorem.

▶ **Theorem 6.** *Let $\Lambda(\boldsymbol{\theta})$ be Lipschitz continuous in the trace distance with constant $\mathcal{L}$. Then $AGF(\Lambda(\boldsymbol{\theta}))$ is Lipschitz continuous with constant $\mathcal{L}$.*

Proof. Recall that

$$\mathrm{AGF}(\Lambda(\boldsymbol{\theta})) := \int \mathrm{d}\psi \langle\psi|\Lambda\left[|\psi\rangle\langle\psi|\right]|\psi|, \rangle \tag{13}$$

so

$$\begin{aligned}
|\mathrm{AGF}(\Lambda(\boldsymbol{\theta})) - \mathrm{AGF}(\Lambda(\boldsymbol{\theta'}))| &= \left| \int \mathrm{d}\psi \langle\psi|\Lambda(\boldsymbol{\theta})[|\psi\rangle\langle\psi|] - \Lambda(\boldsymbol{\theta'})[|\psi\rangle\langle\psi|]|\psi| \right\rangle | && (14)\\
&\leq \int \mathrm{d}\psi \left| \langle\psi|\Lambda(\boldsymbol{\theta})[|\psi\rangle\langle\psi|] - \Lambda(\boldsymbol{\theta'})[|\psi\rangle\langle\psi|]|\psi| \right\rangle | && (15)\\
&\leq \int \mathrm{d}\psi \left\| \Lambda(\boldsymbol{\theta})[|\psi\rangle\langle\psi|] - \Lambda(\boldsymbol{\theta'})[|\psi\rangle\langle\psi|] \right\|_{\mathrm{Tr}} && (16)\\
&\leq \int \mathrm{d}\psi \, \mathcal{L}\|\boldsymbol{\theta} - \boldsymbol{\theta'}\| && (17)\\
&= \mathcal{L}\|\boldsymbol{\theta} - \boldsymbol{\theta'}\|. && \blacktriangleleft
\end{aligned}$$

As noted in the introduction, we do not have direct access to $\mathrm{AGF}(\Lambda_T(\boldsymbol{\theta}))$, but rather to $\mathrm{AGF}(\Lambda_T(\boldsymbol{\theta})\Lambda_{\mathrm{ref}}(\boldsymbol{\theta}))$. In particular, $F(\boldsymbol{\theta}) := \mathrm{AGF}(\Lambda_T(\boldsymbol{\theta})\Lambda_{\mathrm{ref}}(\boldsymbol{\theta}))$ may be estimated from the interleaved randomized benchmarking parameters:

$$p(\boldsymbol{\theta}) := \frac{dF(\boldsymbol{\theta}) - 1}{d - 1}, \tag{18a}$$

$$A(\boldsymbol{\theta}) := \mathrm{Tr}(E\Lambda_{\mathrm{ref}}(\boldsymbol{\theta})[\rho - \frac{\mathbb{1}}{d}]), \tag{18b}$$

$$\text{and } B(\boldsymbol{\theta}) := \mathrm{Tr}(E\Lambda_{\mathrm{ref}}(\boldsymbol{\theta})[\frac{\mathbb{1}}{d}]), \tag{18c}$$

where $d = \dim(\mathcal{H})$, ρ is the state prepared at the start of each sequence, and E is the measurement at the end of each sequence. We consider A and B later, but note for now that up to a factor of $d/(d-1)$, Lipschitz continuity of $F(\boldsymbol{\theta})$ immediately implies Lipschitz continuity of $p(\boldsymbol{\theta})$. Thus, we can follow the same argument as above, but using the channel $\Lambda_T(\boldsymbol{\theta})\Lambda_{\mathrm{ref}}(\boldsymbol{\theta})$ instead to argue the Lipschitz continuity of experimentally accessible estimates.

We proceed to show the Lipschitz continuity of F and hence of p by revisiting the definition Equation 7 of Λ_{ref}. In particular, we partition the twirling group as $G = \bigcup_{n=0}^{\infty} G_n$, where G_n is the set of elements of G whose decomposition into generators $\{T, V_1, \ldots, V_{\ell-1}\}$

requires at least n instances of the target gate T. For instance, if $G = \langle S, H \rangle$ and the target gate is $T = S$, then $Z \in G_2$ since $Z = SS$ is the decomposition of Z requiring the least copies of S. The partition of G in this example is shown as Table 1.

Using this partitioning of G, we can define an analogous partition on the terms occuring in the definition of $\Lambda_{\text{ref}}(\boldsymbol{\theta})$,

$$\Lambda_{\text{ref}}(\boldsymbol{\theta}) = \sum_{n=0}^{\infty} \frac{|G_n|}{|G|} \Lambda_{\text{ref},n}(\boldsymbol{\theta}), \tag{19}$$

$$\text{where } \Lambda_{\text{ref},n}(\boldsymbol{\theta}) := \frac{1}{|G_n|} \sum_{U \in G_n} \Lambda_U(\boldsymbol{\theta}). \tag{20}$$

▶ **Theorem 7.** *If $\Lambda_T(\boldsymbol{\theta})$ is Lipschitz continuous in the trace distance with constant $\mathcal{L}$, then $\Lambda_{\text{ref},n}(\boldsymbol{\theta})$ is Lipschitz continuous in the trace distance with constant nL. Furthermore $\Lambda_{\text{ref}}(\boldsymbol{\theta})$ is Lipshitz continuous with constant $\bar{n} := \sum_{n=0}^{\infty} n \frac{|G_n|}{|G|}$.*

Proof. Consider one of the summands from Equation 20, and without loss of generality let $U = V_{i_0} V_{i_1} \cdots V_{i_k}$ for the sequence of integer indices $\boldsymbol{i} = (i_0, i_1, \ldots, i_k)$. Then, by Equation 5,

$$\Lambda_U(\boldsymbol{\theta}) = \Lambda_{V_{i_0}}(\boldsymbol{\theta})(V_{i_0} \bullet) \cdots \Lambda_{V_{i_k}}(\boldsymbol{\theta})(V_{i_k} \bullet)(U^\dagger \bullet). \tag{21}$$

Note that, $\forall i$, $V_i \perp\!\!\!\perp \boldsymbol{\theta}$ since these are ideal channels and hence independent of the control vector $\boldsymbol{\theta}$; these channels are Lipschitz continuous in the trace distance with constant 0. Further, each $\Lambda_{V_i} \perp\!\!\!\perp \boldsymbol{\theta}$ for $i > 0$; these channels are also Lipschitz continuous in the trace distance with constant 0. By assumption, we have Λ_{V_0} is Lipschitz continuous in the trace distance with constant $\mathcal{L}$. Hence, each factor in Λ_U is Lipschitz continuous in the trace distance with constant $\mathcal{L}$ or 0, as detailed above.

By Corollary 4, Λ_U is Lipschitz continuous in the trace distance with constant mL, where m counts the number of 0s in $\boldsymbol{i}$ (corresponding to the number of times the target gate occurs in the decomposition of U). By construction, $m \leq n$, so Λ_U is also Lipschitz continuous in the trace distance with constant nL.

Using Lemma 5 to, we now have that $\Lambda_{\text{ref},n}(\boldsymbol{\theta})$ is Lipschitz continuous in the trace distance with constant $\frac{1}{|G_n|} \sum_{U \in G_n} nL = nL$, which is what we wanted to show.

We thus have that $\Lambda_{\text{ref}}(\boldsymbol{\theta})$ is Lipschitz continuous in the trace distance with constant $\bar{n}\mathcal{L}$, wherein

$$\bar{n} := \sum_{n=0}^{\infty} n \frac{|G_n|}{|G|} \tag{22}$$

is the average number of times that the target gate T appears in decompositions of elements of the twirling group G. ◀

▶ **Corollary 8** (Lipschitz continuity of $\Lambda_{\text{ref}}(\boldsymbol{\theta})$)**.** *Let*

$$\bar{n} := \sum_{n=0}^{\infty} n \frac{|G_n|}{|G|} \tag{23}$$

be the average number of times that the target gate V_0 appears in decompositions of elements of the twirling group G. Then, $\Lambda_{\text{ref}}(\boldsymbol{\theta})$ is Lipschitz continuous in the trace distance with constant $\bar{n}\mathcal{L}$.

Combining with the previous argument, we thus have our central theorem.

▶ **Theorem 9.** *Let $\Lambda_T(\boldsymbol{\theta})$ be Lipschitz continuous in the trace distance with constant $\mathcal{L}$. Then, $F(\boldsymbol{\theta}) = AGF(\theta)$ is Lipschitz continuous with constant $(1+\bar{n})\mathcal{L}$, and $p(\boldsymbol{\theta})$ is Lipschitz continuous with constant $d(1+\bar{n})\mathcal{L}/(d-1)$, and $A(\boldsymbol{\theta})$ and $B(\boldsymbol{\theta})$ are Lipschitz continuous with constant $\bar{n}\mathcal{L}$.*

Proof. First, $F(\boldsymbol{\theta}) = \mathrm{AGF}(\Lambda_T(\boldsymbol{\theta})\Lambda_{\mathrm{ref}}(\boldsymbol{\theta}))$. By assumption, $\Lambda_T(\boldsymbol{\theta})$ is Lipschitz continuous with constant $\mathcal{L}$, and by Corollary 8, $\Lambda_{\mathrm{ref}}(\boldsymbol{\theta}))$ is Lipschitz continuous with constant $\bar{n}\mathcal{L}$. Hence, by Theorem 6, $F(\boldsymbol{\theta})$ is Lipschitz continuous with constant $(1+\bar{n})\mathcal{L}$.

Next, recall that $p(\boldsymbol{\theta}) = \frac{dF(\boldsymbol{\theta})-1}{d-1}$. Then, it follows that $p(\boldsymbol{\theta})$ is Lipschitz continuous with constant $\frac{d(1+\bar{n})\mathcal{L}}{d-1}$.

For $B(\boldsymbol{\theta})$, we have

$$|B(\boldsymbol{\theta}') - B(\boldsymbol{\theta})| = \left|\mathrm{Tr}(E\Lambda_{\mathrm{ref}}(\boldsymbol{\theta}')[\mathbb{1}/d]) - \mathrm{Tr}(E\Lambda_{\mathrm{ref}}(\boldsymbol{\theta})[\mathbb{1}/d])\right| \tag{24}$$

$$= \left|\mathrm{Tr}(E(\Lambda_{\mathrm{ref}}(\boldsymbol{\theta}') - \Lambda_{\mathrm{ref}}(\boldsymbol{\theta}))[\mathbb{1}/d])\right|. \tag{25}$$

Letting $(\epsilon_0, \epsilon_1, ..., \epsilon_d)$ be the ordered singular values of E and $(\lambda_0, \lambda_1, ..., \lambda_d)$ be the ordered singular values of $(\Lambda_{\mathrm{ref}}(\boldsymbol{\theta}') - \Lambda_{\mathrm{ref}}(\boldsymbol{\theta}))[\mathbb{1}/d]$, we have

$$|B(\boldsymbol{\theta}') - B(\boldsymbol{\theta})| \leq \sum_{i=1}^{d} \epsilon_i\lambda_i \leq \max(\epsilon)\sum_{i=1}^{d}\lambda_i = \max(\epsilon)\|(\Lambda_{\mathrm{ref}}(\boldsymbol{\theta}') - \Lambda_{\mathrm{ref}}(\boldsymbol{\theta}))[\mathbb{1}/d]\|_{\mathrm{Tr}} \leq \bar{n}\mathcal{L}, \tag{26}$$

Since E and C are both Hermitian, EC is also Hermitian, and thus $\|EC\|_{\mathrm{Tr}} = \mathrm{Tr}(|EC|) \geq |\mathrm{Tr}(EC)|$. The argument is completed by Hölder's inequality [36], which states that for all X and Y, $\|XY\|_{\mathrm{Tr}} \leq \|X\|_{\mathrm{Tr}}\|Y\|_{\mathrm{spec}}$, where $\|\cdot\|_{\mathrm{spec}}$ is the spectral norm (*a.k.a.* the induced $(2\to 2)$-norm or Schatten ∞-norm). In particular, we note that since E is a POVM effect, $\|E\|_{\mathrm{spec}} \leq 1$, such that $\|EC\|_{\mathrm{Tr}} \leq \|C\|_{\mathrm{Tr}} \leq 1$.

Finally, we note that this argument goes identically for the state $\rho - \frac{\mathbb{1}}{d}$, as we did not use any special properties of $\frac{\mathbb{1}}{d}$. Hence, we also have that $|A(\boldsymbol{\theta}') - A(\boldsymbol{\theta})| \leq \bar{n}\mathcal{L}$. ◀

We are thusly equipped to return to the problem of estimating $F(\boldsymbol{\theta}+\boldsymbol{\delta\theta})$ from experimental data concerning $F(\boldsymbol{\theta})$.

▶ **Theorem 10.** *Suppose that $f(\boldsymbol{\theta}, \boldsymbol{y})$ is a Lipschitz continuous function of $\boldsymbol{\theta}$ with constant $\mathcal{L}$ where y is a variable in a measurable set S with corresponding probability distribution on that set of $\Pr(\boldsymbol{y})$ and for any function $g : S \mapsto \mathbb{R}$ define $\mathbb{E}_{\boldsymbol{y}}(g(\boldsymbol{y})) = \int_S g(\boldsymbol{y})\Pr(\boldsymbol{y})\mathrm{d}\boldsymbol{y}$ and $Var_{\boldsymbol{y}}(g(\boldsymbol{y})) = \mathbb{E}_{\boldsymbol{y}}\big(g(\boldsymbol{y}) - \mathbb{E}_{\boldsymbol{y}}(g(\boldsymbol{y})\big)^2$. For all $\boldsymbol{\theta}$ and $\boldsymbol{\theta}'$ such that $\mathcal{L}\|\boldsymbol{\theta}' - \boldsymbol{\theta}\| < \sqrt{Var_{\boldsymbol{y}}(f(\boldsymbol{\theta}, \boldsymbol{y}))}$, it holds that*

$$Var_{\boldsymbol{y}}[f(\boldsymbol{\theta}', \boldsymbol{y})] \leq Var_{\boldsymbol{y}}[f(\boldsymbol{\theta}, \boldsymbol{y})]\left(1 + \frac{2\mathcal{L}\|\boldsymbol{\theta}' - \boldsymbol{\theta}\|}{\sqrt{Var_{\boldsymbol{y}}[f(\boldsymbol{\theta}, \boldsymbol{y})]}}\right). \tag{27}$$

Proof. Note that since f is Lipschitz continuous as a function of $\boldsymbol{\theta}$,

$$\left|f(\boldsymbol{\theta}', \boldsymbol{y}) - f(\boldsymbol{\theta}, \boldsymbol{y})\right| \leq \mathcal{L}\|\boldsymbol{\theta}' - \boldsymbol{\theta}\|, \tag{28}$$

so there exists a function c such that $|c(\boldsymbol{\theta}, \boldsymbol{\theta}', \boldsymbol{y})| \leq 1$ for all $\boldsymbol{\theta}$, $\boldsymbol{\theta}'$ and $\boldsymbol{y}$:

$$f(\boldsymbol{\theta}', \boldsymbol{y}) = f(\boldsymbol{\theta}, \boldsymbol{y}) + \mathcal{L}\|\boldsymbol{\theta}' - \boldsymbol{\theta}\|c(\boldsymbol{\theta}, \boldsymbol{\theta}', \boldsymbol{y}). \tag{29}$$

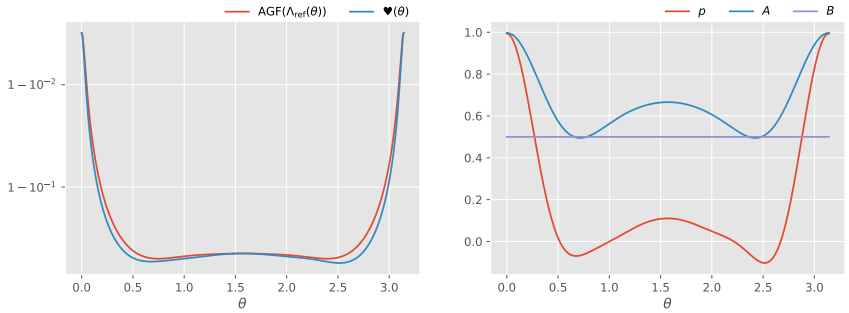


Figure 1 The objective function $F(\theta)$ and the average gate fidelity versus the overrotation angle θ for Example 11 is given in the left figure. The right figure gives the calculated RB parameters as a function of θ where the optimal solution $\theta = 0$ is unknown to the optimizer a priori.

Thus, $\mathrm{Var}_{\boldsymbol{y}}[c] \leq 1$, and by addition of variance, we have that

$$\begin{aligned}\mathrm{Var}_{\boldsymbol{y}}[f(\boldsymbol{\theta}', \boldsymbol{y})] &= \mathrm{Var}_{\boldsymbol{y}}[f(\boldsymbol{\theta}, \boldsymbol{y})] + \mathcal{L}^2\|\boldsymbol{\theta} - \boldsymbol{\theta}'\|^2 \mathrm{Var}_{\boldsymbol{y}}(c(\boldsymbol{\theta}, \boldsymbol{\theta}', \boldsymbol{y})) \\ &\quad + 2\mathcal{L}\|\boldsymbol{\theta} - \boldsymbol{\theta}'\| \mathrm{Cov}_{\boldsymbol{y}}(f(\theta, \boldsymbol{y}), c(\theta, \theta', \boldsymbol{y})) \\ &\leq \mathrm{Var}_{\boldsymbol{y}}[f(\boldsymbol{\theta}, \boldsymbol{y})] + \mathcal{L}^2\|\boldsymbol{\theta} - \boldsymbol{\theta}'\|^2 + \mathcal{L}\|\boldsymbol{\theta} - \boldsymbol{\theta}'\| \sqrt{\mathrm{Var}_{\boldsymbol{y}}(f(\boldsymbol{\theta}, \boldsymbol{y})}. \\ &\leq \mathrm{Var}_{\boldsymbol{y}}[f(\boldsymbol{\theta}, \boldsymbol{y})] + 2\mathcal{L}\|\boldsymbol{\theta} - \boldsymbol{\theta}'\| \sqrt{\mathrm{Var}_{\boldsymbol{y}}(f(\boldsymbol{\theta}, \boldsymbol{y})}. \end{aligned} \tag{30}$$

The result then follows from elementary algebra. ◀

2.1 Examples

▶ **Example 11** (Lipschitz Continuity of Unitary Overrotation)**.** Consider $G = \langle S, H \rangle$, where $T = S$ is the target gate. For a control parameter vector consisting of a single overrotation parameter $\boldsymbol{\theta} = (\delta\theta)$, suppose that $\Lambda_T[\rho] = (e^{-i\,\delta\theta\,\sigma_z}) \bullet \rho$. Since this is a unitary channel, its Choi–Jamiłkowski rank[1] is 1. Thus, the AGF of Λ_T can be calculated as the trace [28, 21, 8]

$$\mathrm{AGF}(\Lambda_T(\delta\theta)) = \frac{|\mathrm{Tr}(e^{-i\,\delta\theta\,\sigma_z})|^2 + 2}{4 + 2} = \frac{2}{3} + \frac{1}{3}\cos(2\,\delta\theta). \tag{31}$$

On the other hand, $F(\delta\theta)$ isn't as straightforward, and so we will consider its Lipschitz continuity instead. To do so, we note that for all $\rho \in \mathrm{D}(\mathbb{C}^2)$, we wish to bound the trace norm

$$\Delta = \|\Lambda_T(\delta\theta)[\rho] - \Lambda_T(\delta\theta')[\rho]\|_{\mathrm{Tr}}. \tag{32}$$

[1] Sometimes informally called a "Kraus rank."

Expanding ρ in the unnormalized Pauli basis as $\rho = \mathbb{1}/2 + \boldsymbol{r}\cdot\boldsymbol{\sigma}/2$, we note that since $\Lambda_T(\delta\theta)[\mathbb{1}] = \mathbb{1}$ and $\Lambda_T(\delta\theta)[\sigma_z] = \sigma_z$ for all $\delta\theta$, the above becomes

$$\begin{aligned}
\Delta &= \frac{1}{2}\|\Lambda_T(\delta\theta)[r_x\sigma_x + r_y\sigma_y + r_z\sigma_z] - \Lambda_T(\delta\theta')[r_x\sigma_x + r_y\sigma_y + r_z\sigma_z]\|_{\mathrm{Tr}} && (33)\\
&= \frac{1}{2}\|\Lambda_T(\delta\theta)[r_x\sigma_x + r_y\sigma_y] - \Lambda_T(\delta\theta')[r_x\sigma_x + r_y\sigma_y]\|_{\mathrm{Tr}} && (34)\\
&= 4|\sin(\delta\theta - \delta\theta')|\sqrt{r_x^2 + r_y^2} && (35)\\
&\leq 4|\sin(\delta\theta - \delta\theta')| && (36)\\
&\leq 4|\delta\theta - \delta\theta'|, && (37)
\end{aligned}$$

where the last line follows from that $|\sin(x)| \leq |x|$. Thus, we conclude that Λ_T is Lipschitz continuous in the trace distance with constant 4.

We can then find $\bar{n}$ for occurrences of T in decompositions of elements of G to find the Lipschitz constant for $F(\delta\theta)$ in this example. In particular, as shown in the Supplementary Material, $\bar{n} = 13/6$ for the presentation of the Clifford group under consideration, such that F is Lipschitz continuous with constant $(d/(d-1)) \times 4 \times (19/6) = 76/3$ in this case.

We note that a more detailed analysis of the Lipschitz continuity of Λ_T or a presentation of G that is less dense in T would both yield smaller Lipschitz constants for F, and hence better reuse of prior information. Thus by Theorem 9, a change in overrotation of approximately $1/100$ the current standard deviation in F would result in at most a doubling of the current standard deviation.

We can easily include the effects of noise in other generators in numerical simulations. In particular, suppose that Λ_H is a depolarizing channel with strength 0.5%. Then, simulating $F(\boldsymbol{\theta})$ for this case shows that F is Lipschitz continuous with a constant of approximately 1.48, as illustrated in Figure 1.

3 Approximate Bayesian Inference

An important implication of Theorem 10 is that the uncertainty quantified by the variance of the posterior distribution yielded by Bayesian inference grows by at most a constant factor. However, while the theorem specify how the variance should grow in the worst case scenario it does not give us an understanding of what form the posterior distribution should take. Our goal in this section is to provide an operationally meaningful way to think about how the posterior distribution evaluated at $\boldsymbol{\theta}$ changes as the control parameters transition to $\boldsymbol{\theta'}$.

Let the posterior probability distribution for the objective function F evaluated at parameters $\boldsymbol{\theta}$ be $\Pr(F(\boldsymbol{\theta}))$. In practice, we do not generally estimate the objective function F directly, but estimate F from a latent variable $\boldsymbol{y}$, such as the RB parameters Equation 18. Marginalizing over this latent variable, we obtain the Bayesian mean estimator for F,

$$\hat{F} = \int F \Pr(F|\theta)\,\mathrm{d}F = \int F \Pr(F|\theta,\boldsymbol{y}) \Pr(\boldsymbol{y})\mathrm{d}\boldsymbol{y}. \tag{38}$$

For the RB case in particular, the objective function F does not depend on the control parameters $\boldsymbol{\theta}$ if we know the RB parameters $\boldsymbol{y}$ exactly. That is, we write that $F \perp\!\!\!\perp \boldsymbol{\theta}|\boldsymbol{y}$ for the RB case, such that $\Pr(F|\boldsymbol{\theta},\boldsymbol{y}) = \Pr(F|\boldsymbol{y})$. Moreover, $\Pr(F|\boldsymbol{y})$ is a δ-distribution supported only at $F = (dp+1)/(d+1)$ where $\boldsymbol{y} = (p, A, B)$. We may thus abuse notation slightly and write that $F = F(\boldsymbol{y})$ is a deterministic function. Doing so, our estimator simplifies considerably, such that

$$\hat{F} = \int F \Pr(F|\theta,\boldsymbol{y}) \Pr(\boldsymbol{y})\mathrm{d}\boldsymbol{y} = \int F(\boldsymbol{y}) \Pr(\boldsymbol{y})\mathrm{d}\boldsymbol{y}. \tag{39}$$

In exact Bayesian inference, the probability density $\Pr(\boldsymbol{y})$ is an arbitrary distribution, but computation of the estimator Equation 39 is in general intractable. Perhaps the most easily generalizable distribution is the sequential Monte Carlo (SMC) approximation [6], also known as a particle filter, which attempts to approximate the probability density as

$$\Pr(F|\theta,\boldsymbol{y})\Pr(\boldsymbol{y}|\boldsymbol{\theta}) = \Pr(F,\boldsymbol{y}|\boldsymbol{\theta}) \approx \sum_{j=1}^{N_p} w_j \delta(\boldsymbol{y}-\boldsymbol{y}_j)\delta(F_i - F), \tag{40}$$

where δ is the Dirac-delta distribution and $\sum_j w_j = 1$. This representation is convenient for recording on a computer, as it only needs to store $(w_i, \boldsymbol{y}_i, F_i)$ for each particle. If $F = F(\boldsymbol{y})$ is a deterministic function of the RB parameters then we need not even record F with each particle, such that

$$\Pr(F|\theta,\boldsymbol{y})\Pr(\boldsymbol{y}|\boldsymbol{\theta}) \approx \sum_{j=1}^{N_p} w_j \delta(\boldsymbol{y}-\boldsymbol{y}_j)\delta(F(\boldsymbol{y}) - F). \tag{41}$$

More generally, the SMC approximation allows us to approximate expectation values over the probability distribution using a finite number of points, or particles, such that the expectation value of any continuous function can be approximated with arbitrary accuracy as $N_p \to \infty$. In particular, we can approximate the estimator $\hat{F}$ within arbitrary accuracy.

The uncertainty (mean squared error) of this estimator is given by the posterior variance,

$$\mathbb{V}(F) = \int F^2 \Pr(F|\theta,\boldsymbol{y})\Pr(\boldsymbol{y})\mathrm{d}\boldsymbol{y} - \hat{F}^2. \tag{42}$$

The posterior variance can be computed as the variance over the variable $\boldsymbol{y}$ induced from the sequential Monte Carlo approximation to the probability distribution,

$$\mathbb{V}(F) \approx \sum_i w_i F(\boldsymbol{y}_i)^2 - \left(\sum_i w_i F(\boldsymbol{y}_i)\right)^2, \tag{43}$$

where we have assumed that $F \perp\!\!\!\perp \boldsymbol{\theta}|\boldsymbol{y}$ and that $\Pr(F|\boldsymbol{\theta})$ is a δ-distribution, as in the RB case. This observation is key to our implementation of Bayesian ACRONYM tuning.

A final note regarding approximate Bayesian inference is that the learning process can be easily implemented. From Equation 1 if $\Pr(F|\theta,\boldsymbol{y})\Pr(\boldsymbol{y}) = \sum_{j=1}^{N_p} w_j \delta(\boldsymbol{y}-\boldsymbol{y}_j)$ and if evidence E is obtained in an experiment, then Bayes' theorem when applied to the weights w_j yields

$$w_j \leftarrow \frac{\Pr(E|\boldsymbol{y_j})w_j}{\sum_j \Pr(E|\boldsymbol{y_j})w_j}. \tag{44}$$

This update procedure is repeated iteratively over all data that is collected from a set of experiments. In practice, if an accurate estimate is needed then an enormous number of particles may be needed because the weights shrink exponentially with the number of updates. This causes the effective number of particles in the approximation to shrink exponentially and with it the accuracy of the approximation to the posterior. We can address this by moving the particles to regions of high probability density. In practice, we use a method proposed by [24] to move the particles but other methods exist and we recommend reviewing [6, 15, 20] for more details. Here, we will use the implementation of particle filtering and Liu–West resampling provided by the QInfer package [14].

3.1 Reusing Priors from Nearby Experiments

We have argued above that the posterior variance of the probability distribution is Lipshitz continuous, which allows us to reason that the variance of the probability distribution at most expands by a fixed multiplicative constant when transitioning information between different points. Operationally though, it is less clear how we should choose the posterior distribution over the average gate fidelity in Bayesian ACRONYM training given prior information at a single point. Theorem 9 provides us with an intuition that can be used for this: each element in the support of the probability distribution is shifted by at most a fixed amount that is dictated by the Lipshitz constants for the channels. Here, we build on this intuition by showing that the prior at each step in a Bayesian ACRONYM tuning protocol can be related to the previous step in terms of the Minkowski sum and convex hull.

▶ **Definition 12** (Convex hull)**.** *Let A be a set of vectors. Then the convex hull of A, written* $\mathrm{Conv}(A)$ *is the smallest convex set containing A,*

$$\mathrm{Conv}(A) := \{\lambda \boldsymbol{a} + (1-\lambda)\boldsymbol{b} : \boldsymbol{a}, \boldsymbol{b} \in A, 0 \leq \lambda \leq 1\}. \tag{45}$$

▶ **Definition 13** (Minkowski sum)**.** *Let A and B be sets of vectors. Then the Minkowski sum $A + B$ is defined as the convolution of A with B,*

$$A + B := \{\boldsymbol{a} + \boldsymbol{b} : \boldsymbol{a} \in A, \boldsymbol{b} \in B\}. \tag{46}$$

With these concepts in place we can now state the following Corollary, which can be used to define a sensible prior distribution for $\boldsymbol{y}(\boldsymbol{\theta} + \boldsymbol{\delta\theta})$ given a posterior distribution for $\boldsymbol{y}(\boldsymbol{\theta})$.

▶ **Corollary 14.** *Let $\Lambda_T(\boldsymbol{\theta})$ be Lipshitz continuous in the trace distance with constant $\mathcal{L}$, and let $\Pr(\boldsymbol{y}|\boldsymbol{\theta})$ be a probability distribution over the RB parameters $\boldsymbol{y} = (p, A, B)$ for Λ_T evaluated at some particular $\boldsymbol{\theta}$. Then, for any $\boldsymbol{\delta\theta} \in \mathbb{R}^n$, let*

$$\Delta := \|\boldsymbol{\delta\theta}\|, \tag{47}$$

$$D := \left\{ \pm \Delta \frac{d\mathcal{L}(1+\bar{n})}{d-1} \right\} \times \left\{ \pm \Delta(1+\bar{n})\mathcal{L} \right\} \times \left\{ \pm \Delta(1+\bar{n})\mathcal{L} \right\}, \tag{48}$$

$$\textit{and } \Pr(\boldsymbol{y}|\boldsymbol{\theta} + \boldsymbol{\delta\theta}) := \frac{1}{8} \sum_{\boldsymbol{s} \in S} \Pr(\boldsymbol{y} - \boldsymbol{s}|\boldsymbol{\theta}). \tag{49}$$

The following statements then hold:

1. $\Pr(\boldsymbol{y}|\boldsymbol{\theta} + \boldsymbol{\delta\theta})$ *is a valid prior probability distribution for* $\boldsymbol{y}(\boldsymbol{\theta} + \boldsymbol{\delta\theta})$.
2. $\hat{y} = \int \boldsymbol{y} \Pr(\boldsymbol{y}|\boldsymbol{\theta})\mathrm{d}\boldsymbol{y} = \int \boldsymbol{y} \Pr(\boldsymbol{y}|\boldsymbol{\theta} + \boldsymbol{\delta\theta})\mathrm{d}\boldsymbol{y}$.
3. *If* $\Pr(\boldsymbol{y}|\boldsymbol{\theta})$ *has support only on* $A \subset \mathbb{R}^3$*, then* $\Pr(\boldsymbol{y}|\boldsymbol{\theta} + \boldsymbol{\delta\theta})$ *has support only on* $\mathrm{Conv}(A + D)$.
4. *If* $\boldsymbol{y}_{true}(\theta) \in A$ *then* $\boldsymbol{y}_{true}(\theta + \delta\theta) \in \mathrm{Conv}(A + D)$.

Proof. The proof of the first claim is trivial and follows immediately from the fact that $\Pr(\boldsymbol{y}|\boldsymbol{\theta})$ is a probability distribution. The proof of the second claim is also straightforward. Note that

$$\begin{aligned} \hat{\boldsymbol{y}} := \int \boldsymbol{y} \Pr(\boldsymbol{y}|\boldsymbol{\theta} + \boldsymbol{\delta\theta})\mathrm{d}\boldsymbol{y} &= \frac{1}{8} \int \sum_{\boldsymbol{s} \in \{\boldsymbol{y}\} + D} \boldsymbol{y} \Pr(\boldsymbol{y} - \boldsymbol{s}|\boldsymbol{\theta})\mathrm{d}\boldsymbol{y} \\ &= \frac{1}{8} \int \sum_{\boldsymbol{s} \in \{\boldsymbol{y}\} + D} (\boldsymbol{y} + \boldsymbol{s}) \Pr(\boldsymbol{y}|\boldsymbol{\theta})\mathrm{d}\boldsymbol{y} \\ &= \int \boldsymbol{y} \Pr(\boldsymbol{y}|\boldsymbol{\theta})\mathrm{d}\boldsymbol{y}. \end{aligned} \tag{50}$$

To consider the third claim, let $\boldsymbol{c} = (c_p, c_A, c_B)$ be a vector such that $|c_p| \leq dL(1+\bar{n})/(d-1)$ and $\max\{|c_A|, |c_B|\} \leq \mathcal{L}(1+\bar{n})$. The convex hull $\mathrm{Conv}(D)$ consists of a convex region of identical dimensions. Since the set is convex it then follows that $\boldsymbol{c} \in \mathrm{Conv}(D)$.

Put differently, we can express Definition 1 and Definition 2 in terms of the Minkowski sum, such that

$$\boldsymbol{y}(\Lambda_T(\boldsymbol{\theta} + \boldsymbol{\delta\theta})) \in \mathrm{Conv}\left(\{\boldsymbol{y}(\Lambda_T(\boldsymbol{\theta})\} + D\right). \tag{51}$$

Taking the union over all vectors $\boldsymbol{a}$ in the support of $\Pr(\boldsymbol{y}|\boldsymbol{\theta})$, we obtain that

$$\mathrm{supp}(\boldsymbol{y}|\boldsymbol{\theta} + \boldsymbol{\delta\theta}) \subseteq \mathrm{Conv}\left(\mathrm{supp}(\boldsymbol{y}|\boldsymbol{\theta}) + D\right). \tag{52}$$

From the linearity of convex hulls under Minkowski summation,

$$\mathrm{Conv}(\mathrm{supp}(\boldsymbol{y}|\boldsymbol{\theta}) + D) = \mathrm{Conv}(\mathrm{supp}(\boldsymbol{y}|\boldsymbol{\theta})) + \mathrm{Conv}(D). \tag{53}$$

The fourth and final statement then immediately follows from Equation 53. ◀

This shows that if we follow the above rule to generate a prior distribution for the RB parameters at $\boldsymbol{\theta} + \boldsymbol{\delta\theta}$ then the resultant distribution does not introduce any bias into the current estimate of the parameters, which is codified by the mean of the posterior distribution. We also have that if the true model is within the support of the prior distribution at $\boldsymbol{\theta}$ then it also will be at $\boldsymbol{\theta} + \boldsymbol{\delta\theta}$. This is important because it states that we can use the resulting distribution to give a credible region for the RB parameters. Thus this choice of prior is well justified and furthermore if the measurement process reduces the posterior variance faster than it expands when $\boldsymbol{\theta}$ is updated, it will allow us to get very accurate estimates of the true RB parameters without needing to extract redundant information.

4 Numerical Experiments

The above analysis shows that, under assumptions of Lipshitz continuity of the likelihood function, the posterior distribution found at a given step of the algorithm can be used to provide a prior for the next step. This holds provided that we form a new prior that expands the variance of the posterior distribution.

While the above analysis shows that prior information can be reused in theory, we will now show in practice that this ability to re-use prior information can reduce the information needed to calibrate a simulated quantum device. The Clifford gates in the device, which we take to be the generators of the single-qubit Clifford group, are H and S. We assume that H can be implemented exactly but that S has an over-rotation error such that

$$S(\theta) = e^{-i\theta Z} S, \tag{54}$$

for some value of θ. While this is called an "over-rotation" we make no assumption that $\theta > 0$. We further apply depolarizing noise at a per-gate level to the system with strength 0.005 meaning that we apply the channels

$$\begin{aligned} \Lambda_H &: \rho \mapsto 0.995 H\rho H + 0.005(\mathbb{1}/2), \\ \Lambda_{S(\theta)} &: \rho \mapsto 0.995 e^{-i\theta Z} S\rho S^\dagger e^{i\theta Z} + 0.005(\mathbb{1}/2). \end{aligned} \tag{55}$$

We assume that the user has control over the parameter θ but we do not assume that they know the functional form and thus do not know that setting $\theta = 0$ will yield optimal performance. The goal of our Bayesian ACRONYM algorithm is then to allow the method to discover that $\theta = 0$ yields the optimal performance via local search.

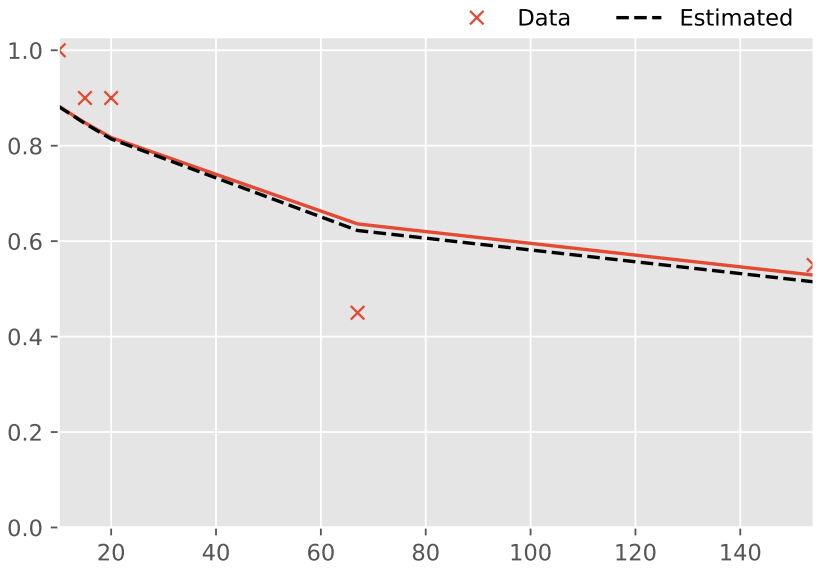


Figure 2 Observed survival probabilities as a function of sequence lengths using 20 measurements (shots) per length for an overrotation model with $\theta = 0.04$. Solid orange line represents the true value for the survival probability, $(A - B)p^{\mathcal{L}} + B$, as a function of the sequence length $\mathcal{L}$ and the dashed line represents the estimate of the survival probability. The prior was set to be uniform for p and A on $[0, 1]$ and the prior B was set to be the normal distribution $\mathcal{N}(0.5, 0.05^2)$.

Figure 2 shows the impact that using Bayesian inference to estimate RB parameters can have in data limited cases of the over-rotation problem. Specifically, we apply Bayesian ACRONYM training to calibrate the over–rotation to within an error of 0.005 which is equal to the dephasing error that we included in the channels in Equation 55. A broad prior was taken and despite the challenges that we would have learning a good model from least-squares fitting, we are able to accurately learn the survival probability. We can then learn the parameters A, B and p, the latter of which gives us the average gate fidelity needed for ACRONYM training via Equation 18a. As the required accuracy for the estimate of p increases, the advantages gleaned from using Bayesian methods relative to fitting disappear [19]. However, in our context this observation is significant because we wish to tune the performance of quantum devices in the small data limit rather than the large data limit and use prior information from previous experiments to compensate.

Local search is implemented using SPSA with learning rate 0.05, a step of 0.05 used to compute approximate gradients and a maximum step size of 0.1. We repeat the method until the posterior variance in the average gate fidelity is less than 0.005^2. We use a Lipshitz constant of 1.48, which was numerically computed as a bound to give an appropriate amount of diffusion for the posterior distribution during an update. Bayesian inference is approximated using a particle filter with 256 000 particles and Liu–West resampling with a resample threshold of 1/256 as implemented by QInfer [14]. Single shot experiments are used with a maximum number of sequences of 500 per set of parameters.

Perhaps the key observation is that throughout the tuning process the true parameters for the overrotation error remain within the 70% credible region reported by QInfer, which suggests if anything that the credible region is pessimistic. The estimate of F also closely tracks the true throughout the learning process and also the amount of data required for the tuning process is minimal, less than 1 kB.

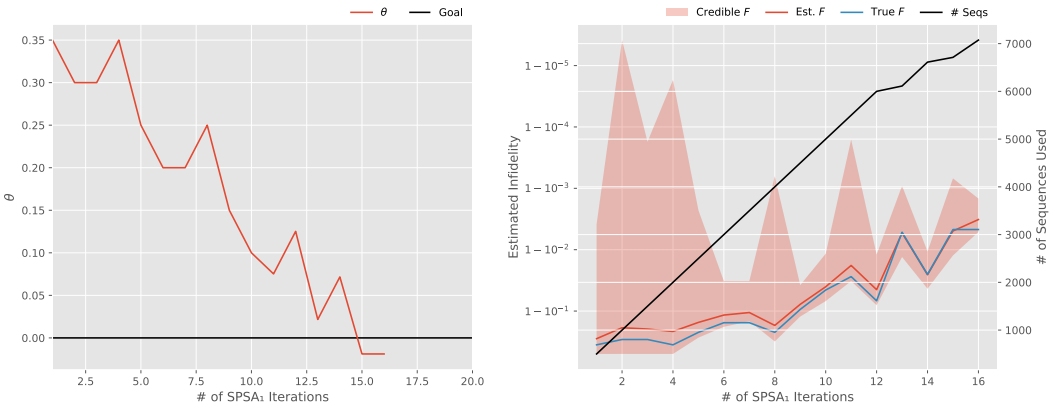


Figure 3 Over-rotation angle and objective function values for an over-rotation model with a 0.35 radian over-rotation initially with a target error of 0.005 in F as measured by the posterior standard-deviation. (Left) Over-rotation angle as a function of number of iterations of SPSA taken. (Right) Estimated Average gate infidelity as a function of the number of SPSA iterations and the total number of sequences used to achieve that level of infidelity. The shaded region represents a 70% credible region for the infidelity.

5 Conclusion

The main result of our work is to show that, under weak assumptions of Lipshitz continuity, Bayesian inference can be used to piece together evidence gained from experiments at nearby experimental settings to accelerate learning of optimal control parameters for quantum devices. We further demonstrate the success of this approach numerically by using a Bayesian ACRONYM tuning protocol (BACRONYM) to tune a rotation gate that suffers from an unknown overrotation. We find that by use of evidence from nearby experimental settings for the gate, we can learn optimal controls with fewer than 1 kilobit of data which is a reduction of nearly a factor of 20 relative to the best known non-Bayesian approach [22].

Looking forward, there are a number of ways in which this work can be built upon. Firstly, upper bounds on the Lipshitz constant and variance are needed to properly use evidence from nearby points within the optimization loop; however, tight estimates are not known a priori for either quantity. Finding approaches that yield useful empirical bounds would be an important contribution beyond what we provide here. Secondly, an experimental demonstration of Bayesian ACRONYM tuning would be useful to demonstrate the viability of such tuning parameters in real-world applications. Finally, while we have picked SPSA as an optimizer for convenience, there may be better choices within the literature. This raises an interesting issue because the number of times that the objective function needs to be queried is not the best metric when information is reused. This point is important not just for choosing the best optimizer to minimize experimental costs for tuning hardware, it also potentially reveals a new way of optimizing parameters in variational quantum eigensolvers [29], as well as QAOA [9] and quantum machine learning algorithms [32].

References

1 Robin Blume-Kohout, John King Gamble, Erik Nielsen, Kenneth Rudinger, Jonathan Mizrahi, Kevin Fortier, and Peter Maunz. Demonstration of qubit operations below a rigorous fault tolerance threshold with gate set tomography. *Nature communications*, 8, 2017.

2 Robert J. Chapman, Christopher Ferrie, and Alberto Peruzzo. Experimental Demonstration of Self-Guided Quantum Tomography. *Physical Review Letters*, 117(4):040402, July 2016. doi:10.1103/PhysRevLett.117.040402.

3 Joshua Combes, Christopher Granade, Christopher Ferrie, and Steven T. Flammia. Logical Randomized Benchmarking. *arXiv:1702.03688 [quant-ph]*, February 2017. arXiv:1702.03688.

4 Andrew W Cross, David P DiVincenzo, and Barbara M Terhal. A Comparative Code Study for Quantum Fault-Tolerance. *arXiv:0711.1556*, November 2007. arXiv:0711.1556.

5 Christoph Dankert, Richard Cleve, Joseph Emerson, and Etera Livine. Exact and Approximate Unitary 2-Designs: Constructions and Applications. *arXiv:quant-ph/0606161*, June 2006. Physical Review A 80, 012304 (2009). arXiv:quant-ph/0606161.

6 Arnaud Doucet and Adam M. Johansen. A Tutorial on Particle Filtering and Smoothing: Fifteen Years Later, 2011.

7 D. J. Egger and F. K. Wilhelm. Adaptive Hybrid Optimal Quantum Control for Imprecisely Characterized Systems. *Physical Review Letters*, 112(24):240503, June 2014. doi:10.1103/PhysRevLett.112.240503.

8 Joseph Emerson, Robert Alicki, and Karol Zyczkowski. Scalable Noise Estimation with Random Unitary Operators. *Journal of Optics B: Quantum and Semiclassical Optics*, 7(10):S347–S352, 2005.

9 Edward Farhi, Jeffrey Goldstone, and Sam Gutmann. A quantum approximate optimization algorithm. *arXiv preprint arXiv:1411.4028*, 2014. arXiv:1411.4028.

10 Christopher Ferrie. Self-Guided Quantum Tomography. *Physical Review Letters*, 113(19):190404, November 2014. doi:10.1103/PhysRevLett.113.190404.

11 Christopher Ferrie and Osama Moussa. Robust and Efficient in Situ Quantum Control. *Physical Review A*, 91(5):052306, May 2015. doi:10.1103/PhysRevA.91.052306.

12 Austin G Fowler, Ashley M Stephens, and Peter Groszkowski. High-threshold universal quantum computation on the surface code. *Physical Review A*, 80(5):052312, 2009.

13 Christopher Granade, Christopher Ferrie, and D. G. Cory. Accelerated Randomized Benchmarking. *New Journal of Physics*, 17(1):013042, January 2015. doi:10.1088/1367-2630/17/1/013042.

14 Christopher Granade, Christopher Ferrie, Ian Hincks, Steven Casagrande, Thomas Alexander, Jonathan Gross, Michal Kononenko, and Yuval Sanders. QInfer: Statistical inference software for quantum applications. *Quantum*, 1:5, 2017.

15 Christopher Granade and Nathan Wiebe. Structured filtering. *New Journal of Physics*, 19(8):083014, 2017.

16 Christopher E Granade. Software Tools for Writing Reproducible Papers. http://www.cgranade.com/blog/2017/05/08/software-for-reproducible-papers.html, May 2017. URL: http://www.cgranade.com/blog/2017/05/08/software-for-reproducible-papers.html.

17 Reinier W. Heeres, Philip Reinhold, Nissim Ofek, Luigi Frunzio, Liang Jiang, Michel H. Devoret, and Robert J. Schoelkopf. Implementing a Universal Gate Set on a Logical Qubit Encoded in an Oscillator. *arXiv:1608.02430 [quant-ph]*, August 2016. arXiv:1608.02430.

18 I. Hincks. personal communications.

19 Ian Hincks, Joel J. Wallman, Chris Ferrie, Chris Granade, and David G. Cory. Bayesian Inference for Randomized Benchmarking Protocols. *arXiv:1802.00401 [quant-ph]*, February 2018. arXiv:1802.00401.

20 Ian Hincks, Joel J Wallman, Chris Ferrie, Chris Granade, and David G Cory. Bayesian Inference for Randomized Benchmarking Protocols. *arXiv preprint arXiv:1802.00401*, 2018. arXiv:1802.00401.

21 Michał Horodecki, Paweł Horodecki, and Ryszard Horodecki. General Teleportation Channel, Singlet Fraction, and Quasidistillation. *Physical Review A*, 60(3):1888–1898, September 1999. doi:10.1103/PhysRevA.60.1888.

22 Julian Kelly, R Barends, B Campbell, Y Chen, Z Chen, B Chiaro, A Dunsworth, Austin G Fowler, I-C Hoi, E Jeffrey, et al. Optimal quantum control using randomized benchmarking. *Physical review letters*, 112(24):240504, 2014.

23 Shelby Kimmel, Marcus P. da Silva, Colm A. Ryan, Blake R. Johnson, and Thomas Ohki. Robust Extraction of Tomographic Information via Randomized Benchmarking. *Physical Review X*, 4(1):011050, March 2014. doi:10.1103/PhysRevX.4.011050.

24 Jane Liu and Mike West. Combined Parameter and State Estimation in Simulation-Based Filtering. In De Freitas and NJ Gordon, editors, *Sequential Monte Carlo Methods in Practice*. Springer-Verlag, New York, 2001.

25 Easwar Magesan, Jay M. Gambetta, and Joseph Emerson. Characterizing Quantum Gates via Randomized Benchmarking. *Physical Review A*, 85(4), April 2012. doi:10.1103/PhysRevA.85.042311.

26 Easwar Magesan, Jay M. Gambetta, B. R. Johnson, Colm A. Ryan, Jerry M. Chow, Seth T. Merkel, Marcus P. da Silva, George A. Keefe, Mary B. Rothwell, Thomas A. Ohki, Mark B. Ketchen, and M. Steffen. Efficient Measurement of Quantum Gate Error by Interleaved Randomized Benchmarking. *Physical Review Letters*, 109(8):080505, August 2012. doi:10.1103/PhysRevLett.109.080505.

27 Seth T Merkel, Jay M Gambetta, John A Smolin, Stefano Poletto, Antonio D Córcoles, Blake R Johnson, Colm A Ryan, and Matthias Steffen. Self-consistent quantum process tomography. *Physical Review A*, 87(6):062119, 2013.

28 Michael A Nielsen. A Simple Formula for the Average Gate Fidelity of a Quantum Dynamical Operation. *quant-ph/0205035*, May 2002. Phys. Lett. A 303 (4): 249-252 (2002). doi:10.1016/S0375-9601(02)01272-0.

29 Alberto Peruzzo, Jarrod McClean, Peter Shadbolt, Man-Hong Yung, Xiao-Qi Zhou, Peter J Love, Alán Aspuru-Guzik, and Jeremy L O'brien. A variational eigenvalue solver on a photonic quantum processor. *Nature communications*, 5:4213, 2014.

30 Timothy Proctor, Kenneth Rudinger, Kevin Young, Mohan Sarovar, and Robin Blume-Kohout. What Randomized Benchmarking Actually Measures. *arXiv:1702.01853 [quant-ph]*, February 2017. arXiv:1702.01853.

31 Łukasz Rudnicki, Zbigniew Puchała, and Karol Zyczkowski. Gauge Invariant Information Concerning Quantum Channels. *arXiv:1707.06926 [quant-ph]*, July 2017. arXiv:1707.06926.

32 Maria Schuld, Alex Bocharov, Krysta Svore, and Nathan Wiebe. Circuit-centric quantum classifiers. *arXiv preprint arXiv:1804.00633*, 2018. arXiv:1804.00633.

33 J.C. Spall. Multivariate Stochastic Approximation Using a Simultaneous Perturbation Gradient Approximation. *IEEE Transactions on Automatic Control*, 37(3):332–341, March 1992. doi:10.1109/9.119632.

34 Joel Wallman, Chris Granade, Robin Harper, and Steven T. Flammia. Estimating the Coherence of Noise. *New Journal of Physics*, 17(11):113020, 2015. doi:10.1088/1367-2630/17/11/113020.

35 Joel J. Wallman. Randomized Benchmarking with Gate-Dependent Noise. *arXiv:1703.09835 [quant-ph]*, March 2017. arXiv:1703.09835.

36 John Watrous. *The Theory of Quantum Information*. Cambridge University Press, Cambridge, United Kingdom, 1 edition edition, April 2018.

A Pseudocode for BACROYNM Tuning

Algorithm 1 Bayesian ACRONYM tuning procedure.

function BACRONYM
$\boldsymbol{\theta}_0$: initial control parameters
n_{shots}: number of measurements per seq. length
σ_{req}: required accuracy for F
(a, b, s, t): SPSA1 parameters
largest allowed step in the parameter $\boldsymbol{\theta}$
F_{target}: target objective function value
π_0: initial prior
$\mathcal{L}$: Lipschitz continuity assumed for F // Initialization
$\pi \leftarrow \pi_0$, $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta}_0$
collect RB data at $\boldsymbol{\theta}$ until $\text{Var}[F] \leq \sigma_{\text{req}}^2$
$\hat{F} \leftarrow \mathbb{E}[F(\boldsymbol{\theta})|\text{data}]$
$i_{\text{iter}} \leftarrow 0$
■ *Main body*
while $\hat{F} \leq F_{\text{target}}$ **do**
i_{iter} ++
■ *SPSA1*
$\boldsymbol{\Delta} \leftarrow$ a random ± 1 vector the same length as $\boldsymbol{\theta}$
step $\leftarrow a/(1 + i_{\text{iter}}^s)$
gain $\leftarrow b/(1 + i_{\text{iter}}^t)$
$\boldsymbol{\delta\theta} \leftarrow \text{step} \cdot \boldsymbol{\Delta}$
estimate $\hat{F}(\boldsymbol{\theta} + \boldsymbol{\delta\theta})$ using Corollary 14
$\boldsymbol{u} \leftarrow \text{gain} \cdot \boldsymbol{\Delta}(\hat{F}(\boldsymbol{\theta} + \boldsymbol{\delta\theta}) - \hat{F}(\boldsymbol{\theta}))$
if any component of $\boldsymbol{u}$ larger than max update **then**
$\boldsymbol{u} \leftarrow \boldsymbol{u}/\max_{u \in \boldsymbol{u}} |u|$
if $|\hat{F}(\boldsymbol{\theta} + \boldsymbol{\delta\theta}) - \hat{F}(\boldsymbol{\theta})| \geq \text{Var}[F(\boldsymbol{\theta} + \boldsymbol{\delta\theta})]$ **then**
$\boldsymbol{\theta}$+= $\boldsymbol{u}$ // Complete the SPSA step.
else if $\hat{F}(\boldsymbol{\theta} + \boldsymbol{\delta\theta}) < \hat{F}(\boldsymbol{\theta})$ **then**
$\boldsymbol{\theta}$−= step $\cdot \boldsymbol{\Delta}$
else
$\boldsymbol{\theta}$+= step $\cdot \boldsymbol{\Delta}$
■ *Final estimate*
return $\boldsymbol{\theta}, \hat{F}$

A Compressed Classical Description of Quantum States

David Gosset
IBM T. J. Watson Research Center, Yorktown Heights, USA
Department of Combinatorics and Optimization and Institute for Quantum Computing, University of Waterloo, Canada
dgosset@uwaterloo.ca

John Smolin
IBM T. J. Watson Research Center, Yorktown Heights, USA
smolin@us.ibm.com

Abstract

We show how to approximately represent a quantum state using the square root of the usual amount of classical memory. The classical representation of an n-qubit state ψ consists of its inner products with $O(\sqrt{2^n})$ stabilizer states. A quantum state initially specified by its 2^n entries in the computational basis can be compressed to this form in time $O(2^n \mathrm{poly}(n))$, and, subsequently, the compressed description can be used to additively approximate the expectation value of an arbitrary observable. Our compression scheme directly gives a new protocol for the *vector in subspace problem* with randomized one-way communication complexity that matches (up to polylogarithmic factors) the optimal upper bound, due to Raz. We obtain an exponential improvement over Raz's protocol in terms of computational efficiency.

2012 ACM Subject Classification Theory of computation → Quantum computation theory; Theory of computation → Quantum complexity theory

Keywords and phrases Quantum computation, Quantum communication complexity, Classical simulation

Digital Object Identifier 10.4230/LIPIcs.TQC.2019.8

Funding We acknowledge support from the IBM Research Frontiers Institute.

Acknowledgements DG thanks Scott Aaronson, Sergey Bravyi, Aram Harrow, Shelby Kimmel, Yi-Kai Liu, Ashley Montanaro, and Oded Regev for helpful discussions. We thank Sergey Bravyi for his Clifford-manipulation code, and Robin Kothari for pointing out the relevance of the lower bound from Ref. [6].

1 Compressing a quantum state

A pure n-qubit state ψ is fully specified by its 2^n amplitudes in the computational basis. Here we are interested in a compressed representation of size $\ll 2^n$ that can be used to approximately recover some of its features.

Aaronson described a compressed representation that can be used to approximate the expectation value of any observable from a known set [1]. He showed that for any n-qubit state ψ and any finite set S of n-qubit observables, there is a compressed representation of ψ of size $O(n \log(n) \log(|S|))$ which suffices to additively approximate expectation values of any observable in S. So for example even if $|S| = \exp(\mathrm{poly}(n))$, then one obtains a remarkable exponential reduction in the classical description size![1] There are two limitations

[1] We note that a different compression method which achieves similar performance to Aaronson's (in terms of the parameters described above) can be inferred from Ref. [3]. In this case the compressed representation is a classical description of the state produced at the output of Algorithm 5.

of Aaronson's scheme that we address here. The first is its efficiency: the algorithm used to compress an n-qubit quantum state scales as $\Omega(c^n|S|)$ for some constant $c > 2$, which can be impractical. A second drawback is that one must first fix the set of observables S and the compressed representation of ψ then depends on this set. In this sense the compressed representation cannot be viewed as an unbiased description of ψ.

Below we present a compressed representation which can be computed quickly, that is, in $O(2^n \mathrm{poly(n)})$ time, given the 2^n amplitudes which fully describe a quantum state ψ. This renders our method practical. Moreover, we do not fix a set of observables a priori–it is possible to approximate the expectation value of an *arbitrary* observable with high probability.

We note that Montanaro has recently ruled out the possibility of a more powerful kind of compressed representation of ψ that would allow one to sample from the probabilty distribution obtained by measuring an observable [12].

A compression scheme of the type we consider has two components. First, there is a classical *compression algorithm* which takes an n-qubit state ψ specified by its amplitudes in the computational basis, along with two parameters $\epsilon, p \in (0, 1)$, and computes the compressed representation which we denote $D(\psi, \epsilon, p)$. The compression algorithm may be probabilistic in which case $D(\psi, \epsilon, p)$ is a random variable. Secondly, there is a classical *expectation value algorithm* which takes as input $D(\psi, \epsilon, p)$ along with an n-qubit Hermitian operator M satisfying $\|M\| \leq 1$, and outputs an estimate E such that

$$|E - \langle\psi|M|\psi\rangle| \leq \epsilon \tag{1}$$

with probability at least $1 - p$, over the randomness used in both the compression and expectation value algorithms. The *classical description size* is the number of bits $|D(\psi, \epsilon, p)|$. We want this to be as small as possible. As we now explain, the best we can hope for is a classical description size which scales mildly exponentially with n.

Indeed, a limitation follows from related work concerning the communication complexity of the *vector in subspace problem* [10, 14, 15]. Suppose Alice can send Bob information over a classical channel and that they may share a random bit string. Alice is given a (classical description) of an n-qubit quantum state ψ and Bob is given an n-qubit projector Π. How much classical information does Alice need to send Bob so that he can compute the expectation value $\langle\psi|\Pi|\psi\rangle$? Suppose that we are promised that either $\langle\psi|\Pi|\psi\rangle \leq 1/3$ or $\langle\psi|\Pi|\psi\rangle \geq 2/3$, so that Bob's goal is to compute just this one bit of information; we allow him to err with probability at most $\frac{1}{4}$, say. The number of bits that Alice needs to send for them to jointly succeed at this task is called the one-way (randomized) classical communication complexity of the vector in subspace problem.

Raz showed that Alice need only send $O(\sqrt{2^n})$ bits for them to succeed [14]. He proposed the following simple protocol which achieves this bound. Using the shared random bit string Alice computes a set of $W = 2^{2^{n/2}}$ Haar random n-qubit states $\phi_1, \phi_2, \ldots, \phi_W$ and then computes the inner products $\langle\phi_1|\psi\rangle, \ldots, \langle\phi_W|\psi\rangle$. She identifies the state ϕ_j such that $|\langle\phi_j|\psi\rangle|$ is maximal and sends the index j to Bob, encoded as a bit string of length $2^{n/2}$. Using the index j along with the shared random string Bob can compute the state ϕ_j. He then computes $\langle\phi_j|\Pi|\phi_j\rangle$ and outputs 1 if it is larger than a certain threshold value and zero otherwise. A detailed analysis of this protocol is provided in Ref. [12]. Note that it is not a practical method of compressing a quantum state as its runtime is doubly exponential in n.

Raz's protocol is optimal in terms of the number of bits communicated. Indeed, a matching lower bound of $\Omega(\sqrt{2^n})$ for the one-way classical communication complexity of the vector in subspace problem can be inferred from the work of Gavinsky et al. [6] (the

following argument was suggested to us by R. Kothari). In that work the authors describe a communication task called the 1/4-Partial Matching problem. Let $N = 2^n$. In this problem Alice is given an N-bit string $x \in \{0,1\}^N$, and Bob is given an $N/4$-bit string $w \in \{0,1\}^{N/4}$ as well as a "partial matching" which is a set of $N/4$ disjoint pairs

$$\{(i_1, j_1), (i_2, j_2), \ldots, (i_{N/4}, j_{N/4})\}$$

such that $1 \leq i_k, j_k \leq N$ for all $k = 1, 2, \ldots, N/4$. We are promised that there exists a bit $b \in \{0,1\}$ such that

$$w_k \oplus x_{i_k} \oplus x_{j_k} = b \qquad 1 \leq k \leq N/4. \tag{2}$$

The goal is to compute the bit b. Gavinsky et al. establish that the classical one-way communication complexity of the 1/4-Partial Matching problem is $\Theta(\sqrt{N})$, and also provide a one-way quantum communication protocol using only $O(\log(N))$ qubits. As we now show, the latter protocol can be recast as a protocol for a specific class of instances of the vector in subspace problem with n qubits. Let us write

$$\mathcal{V} = \text{span}\,\{|i_k\rangle, |j_k\rangle : 1 \leq j, k \leq N/4\}$$

for the $N/2$-dimensional subspace spanned by basis vectors contained in Bob's matching, and let $\mathcal{V}^\perp$ be the orthogonal subspace, which is also $N/2$-dimensional. Finally, let P be a projector onto a subspace of $\mathcal{V}^\perp$ of dimension $N/4$ which is spanned by $N/4$ basis vectors (these can be any $N/4$ basis vectors which are not contained in Bob's matching). Then consider the n-qubit state ψ and projector Π defined as follows.

$$|\psi\rangle = \frac{1}{\sqrt{N}} \sum_{i=1}^{N} (-1)^{x_i} |i\rangle$$

$$\Pi = P + \sum_{k=1}^{N/4} \frac{1}{2} \left(|i_k\rangle - (-1)^{w_k} |j_k\rangle\right) \left(\langle i_k| - (-1)^{w_k} \langle j_k|\right)$$

Using Eq. (2) and the definition of P we see that $\langle\psi|\Pi|\psi\rangle = \frac{1}{4}(1 + 2b)$ for $b \in \{0,1\}$. Thus when $b = 1$ we have $\langle\psi|\Pi|\psi\rangle \geq 2/3$ and when $b = 0$ we have $\langle\psi|\Pi|\psi\rangle \leq 1/3$. This provides the desired reduction from the 1/4-Partial Matching problem with $N = 2^n$ to the vector in subspace problem with n qubits and Ref. [6] then implies a lower bound $\Omega(\sqrt{2^n})$ on its one-way classical communication complexity.

This lower bound directly provides a limitation on compression schemes of the type described above. Indeed, any such compression scheme can be trivially converted into a protocol for solving the vector in subspace problem, with one-way communication complexity $|D(\psi, \epsilon, p)|$ for $\epsilon, p = \Theta(1)$. [2] Thus any compression scheme must have classical description size scaling as $\Omega(\sqrt{2^n})$, so we can only hope to match Raz's upper bound in our slightly more general setting. Finally, although here we will not go beyond the one-way setting, we note that Ref.[15] establishes an exponential lower bound on the total classical communication used to solve the vector in subspace problem allowing for multiple rounds of communication.

In the remainder we present our compression scheme for quantum states. We assume basic knowledge of quantum computation and write $\mathcal{C}_n$ for the n-qubit Clifford group (see, e.g., [7]). Recall that an n-qubit unitary C is an element of $\mathcal{C}_n$ if and only if it can be expressed

[2] Given ψ, Alice computes $D(\psi, \epsilon = 1/100, p = 1/4)$ and sends it to Bob, who then uses it to compute an estimate E such that $|E - \langle\psi|\Pi|\psi\rangle| \leq 1/100$ with probability at least $\frac{1}{4}$. Bob outputs 1 if $E \geq 1/2$ and 0 otherwise.

as a quantum circuit composed of single-qubit Hadamard gates, single-qubit phase gates $S = \mathrm{diag}(1, i)$, and two-qubit CNOT gates. In the following we shall use the fact that (A) the group $\mathcal{C}_n$ of Clifford unitaries forms a unitary 2-design [5], and (B) every Clifford unitary has a short classical description as a quantum circuit consisting of $O(n^2)$ elementary gates.

Let ψ be an n-qubit state. A *stabilizer sketch* of ψ of size 2^k is a classical description of a Clifford unitary $C \in \mathcal{C}_n$ along with the vector of 2^k amplitudes

$$\langle 0^{n-k} z|C|\psi\rangle \qquad\qquad z \in \{0,1\}^k. \tag{3}$$

A random stabilizer sketch of size 2^k is obtained by choosing $C \in \mathcal{C}_n$ uniformly at random. If ψ is stored in computer memory as a list of 2^n amplitudes in the computational basis, then a random stabilizer sketch of ψ can be computed as follows. The random Clifford C can be drawn [9] and expressed as a circuit consisting of $O(n^2)$ one- and two-qubit Clifford gates [2], using a total runtime of $O(n^3)$. This circuit can be applied to ψ to obtain $C|\psi\rangle$ using $O(2^n n^2)$ elementary arithmetic operations. We retain only 2^k computational basis amplitudes of this state along with the classical description of C.

We now show that a handful of random stabilizer sketches of ψ can serve as a compressed representation of ψ. In particular, given parameters $\epsilon, p > 0$, our compressed representation $D(\psi, \epsilon, p)$ consists of $O(\log(p^{-1}))$ independent random stabilizer sketches of ψ, each of size

$$2^k = \tilde{O}(\sqrt{2^n}\epsilon^{-1}) \tag{4}$$

Here and below we use the $\tilde{O}$ notation to hide $\mathrm{poly}(n, \log(\epsilon^{-1}))$ factors, and it is sufficient that the amplitudes Eq.(3) provided in the stabilizer sketches are specified to $\tilde{O}(1)$ bits of precision. We shall also assume $\epsilon \geq 2^{-n/2}$, since otherwise the scaling from Eq. (4) is no better than the trivial $O(2^n)$. Our scheme therefore achieves a classical description size:

$$|D(\psi, \epsilon, p)| = \tilde{O}\big(\sqrt{2^n}\epsilon^{-1}\log(p^{-1})\big), \tag{5}$$

matching Raz and the lower bound from Ref. [6] up to the factors hidden in the $\tilde{O}$. Moreover, the compression algorithm is to simply compute these random stabilizer sketches which takes time $\tilde{O}(2^n)$ as described above.

It remains to exhibit the expectation value algorithm, which takes as input such a compressed representation $D(\psi, \epsilon, p)$ along with an n-qubit observable M and computes an approximation E satisfying Eq. (1) with probability at least $1 - p$. The following lemma describes a slightly more general algorithm which has an improved performance if the observable M has low rank; the claimed expectation value algorithm achieving Eqs. (4,5) is the unrestricted case $r = 2^n$ (note that Eq. (6) matches Eq. (4) whenever $\epsilon \geq 2^{-n/2}$).

▶ **Lemma 1.** *There is a deterministic classical algorithm which takes as input positive integers n, r with $1 \leq r \leq 2^n$ and real numbers $\epsilon, p \in (0, 1)$, along with*

- *An n-qubit Hermitian operator M satisfying $\|M\| \leq 1$ and $\mathrm{rank}(M) \leq r$.*
- *$O(\log(p^{-1}))$ independent random stabilizer sketches of an n-qubit state ψ, each of size 2^k, where k is the largest integer satisfying*

$$2^k \leq 24 \cdot \max\left\{\epsilon^{-2}, \sqrt{r}\epsilon^{-1}\right\}. \tag{6}$$

The algorithm outputs an estimate E such that, with probability at least $1 - p$

$$|E - \langle\psi|M|\psi\rangle| \leq \epsilon.$$

The runtime of the algorithm is $2^{O(n)}\log(p^{-1})$.

This completes our description of the compression scheme for quantum states. We pause for a few remarks before giving the proof below.

As noted above, taking $r = 2^n$ in the Lemma, our compression scheme gives a protocol for the vector in subspace problem. In the opposite extreme case where $r = 1$ we have $M = |\phi\rangle\langle\phi|$ for some n-qubit state ϕ. Here we obtain a one-way communication complexity protocol for approximating the inner product between two vectors ψ (Alice's input) and ϕ (Bob's input) to some additive error ϵ. Choosing $\epsilon = O(1)$, the communication complexity of this protocol is then $\tilde{O}(1)$ which is known to be optimal [11].

The compressed representation can be used to simultaneously approximate expectation values of multiple observables with low probability of failure. Suppose we choose $p = \frac{\delta}{K}$ for some $\delta > 0$ and positive integer K. By a union bound, with probability at least $1 - \delta$, the compressed representation can be used to compute estimates of the expectation values for all observables in any set S of size K, such that all are within the desired approximation error ϵ. Crucially, the compressed representation does not depend on the set S, which highlights the difference between our setting and the one considered by Aaronson [1].

Finally, an interesting special case is when the observable M is a stabilizer projector, i.e.,

$$M = C^\dagger \left(|0\rangle\langle 0|^{\otimes m} \otimes I_{n-m} \right) C$$

for some Clifford $C \in \mathcal{C}_n$ and integer $0 \leq m \leq n$. For example, to approximate the expectation value of any n-qubit Pauli P it suffices to approximate the expected value of the stabilizer projector $(I + P)/2$. As we explain in more detail below, if M is a stabilizer projector then the algorithm from the Lemma has an improved runtime of $\tilde{O}(r \log(p^{-1}))$ and only uses $\tilde{O}(\sqrt{r} \log(p^{-1}))$ space.

We now present the proof of the lemma.

Proof. Note that since k is the largest positive integer satisfying Eq.(6), we have

$$2^k \geq 12 \cdot \max\left\{ \epsilon^{-2}, \sqrt{r}\epsilon^{-1} \right\}. \tag{7}$$

Suppose first that we are given just 2 independent random stabilizer sketches of ψ of size 2^k. The two sketches comes with n-qubit Cliffords $C, D \in \mathcal{C}_n$; we define the associated stabilizer code projectors

$$P = C^\dagger \left(|0\rangle\langle 0|_{n-k} \otimes I_k \right) C \qquad Q = D^\dagger \left(|0\rangle\langle 0|_{n-k} \otimes I_k \right) D. \tag{8}$$

Since C, D are uniformly random, P and Q are uniformly random n-qubit stabilizer projectors of rank 2^k. Therefore

$$\mathbb{E}[P] = \mathbb{E}[Q] = 2^{k-n} I. \tag{9}$$

Using the fact that the n-qubit Clifford group is a unitary 2-design [5] one can also directly show the following fact (we provide a proof for completeness below).

▷ Claim 2.

$$\mathbb{E}[P \otimes P] = \mathbb{E}[Q \otimes Q] = a \cdot I + b \cdot \text{SWAP} \tag{10}$$

where

$$a = \frac{4^{k+n} - 2^{k+n}}{4^{2n} - 4^n} \leq 4^{k-n} \qquad b = \frac{4^n 2^k - 2^n 4^k}{4^{2n} - 4^n} \leq 2^k 4^{-n}. \tag{11}$$

Here and below we write SWAP for the unitary which swaps two n-qubit registers, defined by its action on computational basis states: $\mathrm{SWAP}|x \otimes y\rangle = |y \otimes x\rangle$.

From the two stabilizer sketches we may compute

$$F = 4^{n-k}\mathrm{Re}\left(\langle\psi|PMQ|\psi\rangle\right) \tag{12}$$

$$= 4^{n-k}\mathrm{Re}\Big(\sum_{x,y\in\{0,1\}^k} \langle\psi|C^\dagger|0^{n-k}x\rangle\langle 0^{n-k}x|CMD^\dagger|0^{n-k}y\rangle\langle 0^{n-k}y|D|\psi\rangle\Big). \tag{13}$$

Indeed, Eq. (13) expresses F in terms of the amplitudes $\langle\psi|C^\dagger|0^{n-k}x\rangle$, $\langle 0^{n-k}y|D|\psi\rangle$ from the two given stabilizer sketches as well as matrix elements $\langle 0^{n-k}x|CMD^\dagger|0^{n-k}y\rangle$ which can be computed from the classical descriptions of Cliffords C, D and the given observable M. Note that the computation of F involves a summation of 2^{2k} terms and each term involves the computation of a matrix element $\langle 0^{n-k}x|CMD^\dagger|0^{n-k}y\rangle$, which takes time $2^{O(n)}$. Therefore the total time to compute F given the two stabilizer sketches is $2^{O(n)}$. We note that in the special case where M is a stabilizer projector the matrix element $\langle 0^{n-k}x|CMD^\dagger|0^{n-k}y\rangle$ is equal to the inner product between stabilizer states $\langle 0^{n-k}x|C$ and $MD^\dagger|0^{n-k}y\rangle$ and can be computed in time $O(n^3)$ (see e.g., Ref. [4]). Thus in this case F can be computed using time $\tilde{O}(2^{2k}) = \tilde{O}(r)$ and space $\tilde{O}(\sqrt{r})$.

Using Eqs. (9,12) we see that

$$\mathbb{E}\left[F\right] = \langle\psi|M|\psi\rangle. \tag{14}$$

We now bound the variance of F. First note that

$$4^{-2(n-k)}F^2 \leq |\langle\psi|PMQ|\psi\rangle|^2. \tag{15}$$

We use the identity

$$\mathrm{Tr}\left(\alpha\otimes\gamma\cdot\beta\otimes\delta\cdot\mathrm{SWAP}\right) = \mathrm{Tr}\left(\alpha\beta\gamma\delta\right)$$

which holds for square matrices $\alpha, \beta, \gamma, \delta$ (all of the same dimensions). Using this fact in Eq. (15) we get

$$4^{-2(n-k)}F^2 \leq \left|\mathrm{Tr}\big(\left(|\psi\rangle\langle\psi|\otimes M\right)\left(P\otimes Q\right)\mathrm{SWAP}\big)\right|^2 \tag{16}$$

$$= \mathrm{Tr}\big(\left(|\psi\rangle\langle\psi|\otimes M\otimes M\otimes|\psi\rangle\langle\psi|\right)\left(P\otimes Q\otimes P\otimes Q\right)\mathrm{SWAP}_{12}\mathrm{SWAP}_{34}\big). \tag{17}$$

Now taking the expectation value of both sides and using Eq. (10) we obtain

$$4^{-2(n-k)}\mathbb{E}\left[F^2\right] \leq \mathrm{Tr}\big(\Gamma\cdot\left(|\psi\rangle\langle\psi|\otimes M\otimes M\otimes|\psi\rangle\langle\psi|\right)\big)$$

where

$$\Gamma = \big(a\cdot I + b\cdot\mathrm{SWAP}_{13}\big)\big(a\cdot I + b\cdot\mathrm{SWAP}_{24}\big)\mathrm{SWAP}_{12}\mathrm{SWAP}_{34}$$

and a, b are defined in Eq. (11). Evaluating the trace term by term we arrive at

$$4^{-2(n-k)}\mathbb{E}\left[F^2\right] \leq a^2\left(\langle\psi|M|\psi\rangle\right)^2 + 2ab\langle\psi|M^2|\psi\rangle + b^2\mathrm{Tr}(M^2) \tag{18}$$

$$\leq 4^{-2(n-k)}\left(\langle\psi|M|\psi\rangle\right)^2 + 2\cdot 2^{3k}4^{-2n}\langle\psi|M^2|\psi\rangle + 4^{k-2n}\mathrm{Tr}(M^2) \tag{19}$$

where in the last line we used Eq. (11). Therefore

$$\mathrm{Var}(F) = \mathbb{E}\left[F^2\right] - \big(\mathbb{E}[F]\big)^2 \leq \frac{2}{2^k}\langle\psi|M^2|\psi\rangle + \frac{1}{4^k}\mathrm{Tr}(M^2). \tag{20}$$

Using the bounds

$$\mathrm{Tr}(M^2) \leq \|M\|^2 \mathrm{rank}(M) \leq r,$$

and $\langle\psi|M^2|\psi\rangle \leq \|M\|^2 \leq 1$ in Eq.(20) gives

$$\mathrm{Var}(F) \leq \frac{2}{2^k} + \frac{r}{4^k} \leq \frac{\epsilon^2}{6} + \frac{\epsilon^2}{12^2} \leq \frac{\epsilon^2}{4}, \tag{21}$$

where we used Eq. (7).

Putting together Eqs. (14,21) and applying Chebyshev's inequality we get

$$\Pr\left[|F - \langle\psi|M|\psi\rangle| \geq \epsilon\right] \leq \frac{1}{4}.$$

We have shown that just two stabilizer sketches of size 2^k suffice to compute an estimate F which, with probability at least $3/4$, achieves the desired precision ϵ. The success probability can be amplified by taking the median of multiple independent estimates [8]. That is, now using $2L$ random stabilizer sketches (of the same size 2^k), we compute L independent estimates $F_1, \ldots, F_L$ as above and compute the median

$$E = \mathrm{Median}(F_1, F_2, \ldots, F_L).$$

Note that if $|E - \langle\psi|M|\psi\rangle| \geq \epsilon$ then at least $\frac{1}{2}(L-1)$ of the estimates F_j lie outside the interval $(\langle\psi|M|\psi\rangle - \epsilon, \langle\psi|M|\psi\rangle + \epsilon)$. Since these events are independent and each occurs with probability at most $1/4$ we have

$$\Pr\left[|E - \langle\psi|M|\psi\rangle| \geq \epsilon\right] \leq \sum_{\frac{1}{2}(L-1) \leq \ell \leq L} \binom{L}{\ell} \left(\frac{3}{4}\right)^{L-\ell} \left(\frac{1}{4}\right)^{\ell}. \tag{22}$$

By a Chernoff bound the right-hand side of Eq. (22) can be made $\leq p$ by choosing $L = O(\log(p^{-1}))$.

The algorithm which computes E simply uses Eq. (13) to compute the independent estimates $F_1, \ldots, F_L$ and then takes the median. The computation of each F_j takes time $2^{O(n)}$ as discussed above. ◀

This compression scheme may be useful as a memory-saving means of storing or transmitting the output of classical simulations of large quantum circuits. Suppose the n-qubit output state $|\psi\rangle$ of such a simulation is compressed and sent to a client with at most $O(\sqrt{2^n})$ memory available on her local machine. With this much memory the client cannot store the full state ψ but she can store its compressed representation and use it to approximate the expectation value of any stabilizer projector (as discussed above). The amount of memory savings that could be achieved in practice is determined by the size 2^k of the stabilizer sketches which should be used for a given number of qubits n and error ϵ. Let us suppose for simplicity we are willing to accept a failure probability of $p = 1/4$ so that only 2 stabilizer sketches are needed using the above scheme. To obtain a more precise estimate of this k than the one from Eq. (6), we may solve for $\mathrm{Var}(F) \leq \epsilon^2/4$ using the upper bound on $\mathrm{Var}(F)$ from Eq. (20). For example, for a very large state of $n = 50$ qubits we may choose $k = 35$ to obtain error $\epsilon = 0.0039$. This corresponds to a memory savings of a factor of $2^{14} = 16384$.

Fig. 1 shows the distribution of the estimator F from Eq. (12) in three examples where $n = 12$, $r = 13$, and $k = 7, 8, 9$. In all of these examples the standard deviation of the samples is within a factor of 2 of the upper bound on the standard deviation computed from Eq. (20), showing that this upper bound cannot be improved much further.

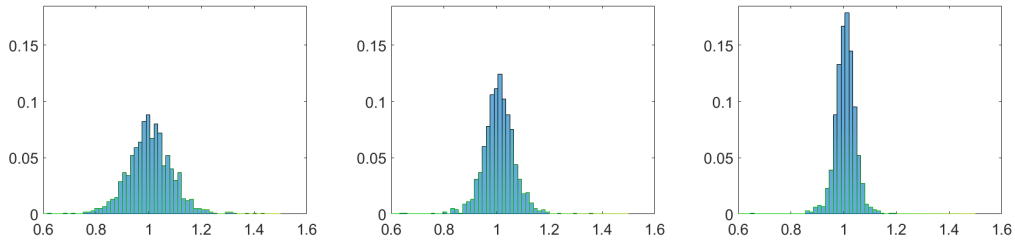


Figure 1 These histograms show the distribution of the random variable F defined in Eq. (12) in three examples where $n = 12$ and $k = 7, 8, 9$ (left to right). Here ψ was chosen to be a random state in the 12-qubit symmetric subspace and Π was chosen to be the projector onto this subspace, which has rank $r = 13$. Thus $\mathbb{E}[F] = \langle\psi|\Pi|\psi\rangle = 1$. For each $k = 7, 8, 9$ we computed 1000 realizations of F. The upper bound on the standard deviation computed from Eq. (20) gives $0.1259, 0.0887, 0.0626$ respectively while the standard deviation of the samples was computed to be $0.0875, 0.0637, 0.0399$. These figures were generated using Python libraries for Clifford manipulations which will soon be available in QISKit[13].

Proof of Claim 2. The n-qubit Clifford group $\mathcal{C}_n$ is a unitary 2-design [5]. This means that if $C \in \mathcal{C}_n$ is uniformly random then for any $2n$-qubit operator ρ we have

$$\mathbb{E}\left[C^\dagger \otimes C^\dagger \rho C \otimes C\right] = \int_{Haar} dU\; U^\dagger \otimes U^\dagger \rho U \otimes U$$

where the right-hand side is integrated over the Haar measure on the unitary group $U(2^n)$. Define projectors $\pi_\pm = (I \pm \mathrm{SWAP})/2$ onto the symmetric and antisymmetric subspaces of two n-qubit registers. Using Schur's lemma the right-hand side of the above expression can be further simplified to

$$\int_{Haar} dU\; U^\dagger \otimes U^\dagger \rho U \otimes U = \pi_+ \frac{\mathrm{Tr}(\rho\pi_+)}{\mathrm{Tr}(\pi_+)} + \pi_- \frac{\mathrm{Tr}(\rho\pi_-)}{\mathrm{Tr}(\pi_-)}.$$

Eq. (10) is obtained by applying this formula in the case $\rho = R \otimes R$, where $R = |0\rangle\langle 0|_{n-k} \otimes I_k$, and substituting

$$\mathrm{Tr}(R \otimes R \cdot \pi_\pm) = \frac{\mathrm{Tr}(R)^2 \pm \mathrm{Tr}(R)}{2} = \frac{2^{2k} \pm 2^k}{2} \qquad \mathrm{Tr}(\pi_\pm) = \frac{4^n \pm 2^n}{2}. \qquad \triangleleft$$

References

1 Scott Aaronson. Limitations of quantum advice and one-way communication. In *Computational Complexity, 2004. Proceedings. 19th IEEE Annual Conference on*, pages 320–332. IEEE, 2004.

2 Scott Aaronson and Daniel Gottesman. Improved simulation of stabilizer circuits. *Physical Review A*, 70(5):052328, 2004.

3 Fernando GSL Brandão, Amir Kalev, Tongyang Li, Cedric Yen-Yu Lin, Krysta M Svore, and Xiaodi Wu. Exponential quantum speed-ups for semidefinite programming with applications to quantum learning. *arXiv preprint*, 2017. `arXiv:1710.02581`.

4 Sergey Bravyi and David Gosset. Improved classical simulation of quantum circuits dominated by Clifford gates. *Physical review letters*, 116(25):250501, 2016.

5 Christoph Dankert, Richard Cleve, Joseph Emerson, and Etera Livine. Exact and approximate unitary 2-designs and their application to fidelity estimation. *Physical Review A*, 80(1):012304, 2009.

6 Dmitry Gavinsky, Julia Kempe, Iordanis Kerenidis, Ran Raz, and Ronald De Wolf. Exponential separations for one-way quantum communication complexity, with applications to cryptography. In *Proceedings of the thirty-ninth annual ACM symposium on Theory of computing*, pages 516–525. ACM, 2007.

7 Daniel Gottesman. The Heisenberg representation of quantum computers. *arXiv preprint*, 1998. `arXiv:quant-ph/9807006`.

8 Mark R Jerrum, Leslie G Valiant, and Vijay V Vazirani. Random generation of combinatorial structures from a uniform distribution. *Theoretical Computer Science*, 43:169–188, 1986.

9 Robert Koenig and John A Smolin. How to efficiently select an arbitrary Clifford group element. *Journal of Mathematical Physics*, 55(12):122202, 2014.

10 Ilan Kremer. *Quantum communication*. Hebrew University of Jerusalem, 1995. (Master's thesis).

11 Ilan Kremer, Noam Nisan, and Dana Ron. On randomized one-round communication complexity. *Computational Complexity*, 8(1):21–49, 1999.

12 Ashley Montanaro. Quantum states cannot be transmitted efficiently classically. *arXiv preprint*, 2016. `arXiv:1612.06546`.

13 QISKit, the quantum information software kit. URL: `https://qiskit.org`.

14 Ran Raz. Exponential separation of quantum and classical communication complexity. In *Proceedings of the thirty-first annual ACM symposium on Theory of computing*, pages 358–367. ACM, 1999.

15 Oded Regev and Bo'az Klartag. Quantum one-way communication can be exponentially stronger than classical communication. In *Proceedings of the forty-third annual ACM symposium on Theory of computing*, pages 31–40. ACM, 2011.

Approximate Unitary $n^{2/3}$-Designs Give Rise to Quantum Channels with Super Additive Classical Holevo Capacity

Aditya Nema
School of Technology and Computer Science, Tata Institute of Fundamental Research, Mumbai, India
aditya.nema30@gmail.com

Pranab Sen
School of Technology and Computer Science, Tata Institute of Fundamental Research, Mumbai, India
http://www.tcs.tifr.res.in/~pgdsen/
pranab.sen.73@gmail.com

Abstract

In a breakthrough, Hastings [10] showed that there exist quantum channels whose classical Holevo capacity is superadditive i.e. more classical information can be transmitted by quantum encoding strategies entangled across multiple channel uses as compared to unentangled quantum encoding strategies. Hastings' proof used Haar random unitaries to exhibit superadditivity. In this paper we show that a unitary chosen uniformly at random from an approximate $n^{2/3}$-design gives rise to a quantum channel with superadditive classical Holevo capacity, where n is the dimension of the unitary exhibiting the Stinespring dilation of the channel superoperator. We do so by showing that the minimum output von Neumann entropy of a quantum channel arising from an approximate unitary $n^{2/3}$-design is subadditive, which by Shor's work [26] implies superadditivity of classical Holevo capacity of quantum channels.

We follow the geometric functional analytic approach of Aubrun, Szarek and Werner [3] in order to prove our result. More precisely we prove a sharp Dvoretzky-like theorem stating that, with high probability under the choice of a unitary from an approximate t-design, random subspaces of large dimension make a Lipschitz function take almost constant value. Such theorems were known earlier only for Haar random unitaries. We obtain our result by appealing to Low's technique [16] for proving concentration of measure for an approximate t-design, combined with a stratified analysis of the variational behaviour of Lipschitz functions on the unit sphere in high dimension. The stratified analysis is the main technical advance of this work.

Haar random unitaries require at least $\Omega(n^2)$ random bits in order to describe them with good precision. In contrast, there exist exact $n^{2/3}$-designs using only $O(n^{2/3}\log n)$ random bits [15]. Thus, our work can be viewed as a partial derandomisation of Hastings' result, and a step towards the quest of finding an explicit quantum channel with superadditive classical Holevo capacity.

2012 ACM Subject Classification Theory of computation → Quantum information theory; Theory of computation → Pseudorandomness and derandomization

Keywords and phrases classical Holevo capacity, super additivity, Haar measure, approximate unitary t-design, polyomial approximation

Digital Object Identifier 10.4230/LIPIcs.TQC.2019.9

Related Version A full version of the paper is available at https://arxiv.org/abs/1902.10808.

1 Introduction

For the past two decades, additivity conjectures have been extensively studied in quantum information theory e.g. [5, 22, 1, 20, 26, 11]. In this paper, we concentrate on the issue of additivity of classical Holevo capacity of a quantum channel Φ, denoted henceforth by $C(\Phi)$.

The quantity $C(\Phi)$ is the number of classical bits of information per channel use that can reliably be transmitted in the limit of infinitely many independent uses of Φ. Capacities of classical memoryless channels are known to be additive, that is, the capacity of two channels Φ and Ψ, used independently, is the sum of the individual capacities. In other words, $C(\Phi \otimes \Psi) = C(\Phi) + C(\Psi)$. This additivity property leads to a single letter characterization of the capacity of classical channels viz. the capacity is nothing but the mutual information between the input and channel output maximised over all possible input distributions for one channel use [27]. For a long time, in analogy with the classical setting, it was generally believed that the classical Holevo capacity of a quantum channel is additive. In fact, this belief was proven to be true for several classes of quantum channels e.g. [13, 8, 12, 25, 14]. Thus, it came as a major surprise to the community when Hastings, in a major breakthrough, showed that there are indeed quantum channels with superadditive classical Holevo capacity [10] i.e. there are quantum channels Φ, Ψ such that $C(\Phi \otimes \Psi) > C(\Phi) + C(\Psi)$.

Hastings' proof proceeds by showing that a Haar random unitary leads to such channels with high probability, in the sense that the unitary, when viewed suitably, is the Stinespring dilation of a quantum channel with superadditive classical Holevo capacity. The drawback of using Haar random unitaries is that they are inefficient to implement. In fact, it takes at least $\Omega(n^2 \log(1/\epsilon))$ random bits in order to pick an $n \times n$ Haar random unitary to within a precision of ϵ in the ℓ_2-distance [28]. Hence, it is of considerable interest to find an explicit efficiently implementable unitary that gives rise to a quantum channel with superadditive classical Holevo capacity.

In this paper, we take the first step in this direction. We show that with high probability a uniformly random $n \times n$ unitary from an approximate $n^{2/3}$-design leads to a quantum channel with superadditive classical Holevo capacity. Though no efficient algorithms for implementing approximate $n^{2/3}$-designs are known, nevertheless, it is known that a uniformly random unitary from an exact $n^{2/3}$-design can be sampled using only $O(n^{2/3} \log n)$ random bits [15, Theorem 3.3]. Also, efficient constructions of approximate $(\log n)^{O(1)}$-designs are known [24, 6]. Thus, our work can be viewed as a partial derandomisation of Hastings' result, and a step towards the quest of finding an explicit quantum channel with superadditive classical Holevo capacity.

Hastings' proof was considerably simplified by Aubrun, Szarek and Werner [3] who showed that existence of channels with subadditive minimum output von Neumann entropy follows from a sharp Dvoretzky-like theorem which states that, under the Haar measure, random subspaces of large dimension make a Lipschitz function take almost constant value. Dvoretzky's original theorem [7] stated that any centrally symmetric convex body can be embedded with low distortion into a section of a high dimensional unit ℓ_2-sphere. Milman [17] extended Dvoretzky's theorem by proving that, with high probability, Haar random subspaces of an appropriate dimension make a Lipschitz function take almost constant value. Dvoretzky's theorem becomes the special case of Milman's theorem where the Lipschitz function happens to be norm induced by the centrally symmetric convex body i.e. the norm under which the convex body becomes the unit ball. Milman's work started a whole body of research sharpening the various parameters of the extended Dvoretzky theorem e.g. [23, 9] etc. However, all these works use Haar random subspaces. A Haar random subspace of $\mathbb{C}^n$ of dimension d can be obtained by applying a Haar random unitary to a fixed subspace of dimension d e.g. the subspace spanned by the first d standard basis vectors of $\mathbb{C}^n$. Our work is the first one to replace the Haar random unitary in any Dvoretzky-type theorem by a uniformly random unitary chosen from an approximate t-design for a suitable value of t. In other words, our main technical result is an Aubrun-Szarek-Werner style

result for approximate t-designs instead of Haar random unitaries. As a corollary, we obtain the subadditivity of minimum output von Neumann entropy for unitaries chosen from an approximate $n^{2/3}$-design. As another corollary, we obtain the subadditivity of minimum output Rényi p-entropy for all $p > 1$ for quantum channels arising from unitaries chosen from approximate unitary $p^3 n^{\frac{4}{9p}+\frac{5}{9}+\frac{2p}{3}} \log n$-design. Such a unitary can in fact be chosen from an exact $p^3 n^{\frac{4}{9p}+\frac{5}{9}+\frac{2p}{3}} \log n$-design using only $p^3 n^{\frac{4}{9p}+\frac{5}{9}+\frac{2p}{3}} (\log n)^2$ random bits [15], which is much less than $\Omega(n^2)$ random bits required to choose a Haar random unitary. Subadditivity of minimum output Rényi p-entropy for all $p > 1$ was originally proved for Haar random unitaries by Hayden and Winter [11].

To prove our main technical result, we use a concentration of measure result by Low [16] for approximate unitary t-designs, combined with a stratified analysis of the variational behaviour of Lipschitz functions on the unit sphere in high dimension. We need such a fine grained stratified analysis for the following reason. Aubrun, Szarek and Werner [3] worked with the function $f(M) := \|MM^\dagger - (I/k)\|_2$, where the argument M is a k^3-tuple rearranged to form a $k \times k^2$ matrix. They found subspaces of dimension k^2 where f took almost constant value. For this, they had to do a two step analysis. The global Lipschitz constant of f was 2 which, under naive Dvoretzky type arguments, would only guarantee the existence of subpaces of dimension $\frac{k^2}{\log k}$ where f is almost constant. This does not suffice to find a counter example to minimum output von Neumann entropy. In order to shave off the $\log k$ term in the denominator, they had to use several sophisticated arguments. One of them was the observation that there is a high probablity subset T of $\mathbb{S}_{\mathbb{C}^{k^3}}$ on which the Lipschitz constant of f was $k^{-1/2}$. They exploited this by their two step analysis, where they separately analysed the behaviour of f on T and on T^c, and managed to shave off the $\log k$ term. For us, since we are working with designs, we need the function to be a polynomial. Hence, instead of f, we have to work with f^2. This seemingly trivial change introduces severe technical difficulties. The main reason behind them is that the Lipschitz constant of f^2 is about twice the Lipschitz constant for f but the variation that we are looking to bound is around square of the earlier variation! This contradiction lies at the heart of the technical difficulty. In order to overcome this, we have to partition $\mathbb{S}_{\mathbb{C}^{k^3}}$ into a number of sets $\Omega_1, \Omega_2, \ldots, \Omega_{\log k}$, called 'layers', with local Lipschitz constants for f^2 running as $k^{-3/2}, 2^3 k^{-3/2}, 3^3 k^{-3/2}, \ldots, (\log k)^3 k^{-3/2}$. We have to bound the variation of f^2 individually on Ω_i as well as put them together to bound the variation on large subspheres of $\mathbb{S}_{\mathbb{C}^{k^3}}$. This leads to a challenging stratified analysis, which forms the main technical advance of this paper.

Another tool developed in this work which should find use in other situations also, is a systematic way to approximate a monotonic differentiable function and its derivative using moderate degree polynomials. This tool is crucially used to prove strict subadditivity of Rényi p-entropy for any $p > 1$ for channels whose unitary Stinespring dilation is chosen from an approximate design instead of a Haar random unitary.

The power of our stratified analysis shows up in the consequence that the dimension of the subspace on which the Lipschitz function is almost constant depends only on the smallest local Lipschitz constant, provided some mild niceness conditions are satisfied. This gives larger dimensional subspaces than a naive analysis which would depend on the global Lipschitz constant. In fact, the stratified analysis allows us to prove a sharper Dvoretzky-type theorem even for the Haar measure. As a result, we can recover Aubrun, Szarek and Werner's result for the function f directly and elegantly instead of applying their Dvoretzky-type result twice which is rather messy. Another powerful consequence of our stratified analysis is that with probability exponentially close to one random, via Haar or t-design, subspaces

make the Lipschitz function almost constant. In contrast, Aubrun, Szarek and Werner could only guarantee constant probability close to one.

The rest of the paper is organised as follows. Section 2 contains notations, symbols definitions and preliminary tools required for the paper. Section 3 states and proves the main technical theorems viz. the stratified analyses for Haar measure and approximate t-designs. Section 4 describes the application to subadditivity of minimum output von Neumann entropy. Section B describes the application to subadditivity of minimum output Rényi p-entropy for $p > 1$. The detailed proofs of all the lemmas and propositions can be found in the full version [19].

2 Preliminaries

All Hilbert spaces used in this paper are finite dimensional. The n dimensional space over complex numbers, $\mathbb{C}^n$, is endowed with the standard inner product aka the dot product: $\langle x, y\rangle := \sum_{i=1}^{n} x_i^* y_i$. The unit radius sphere in $\mathbb{C}^n$ is denoted by $\mathbb{S}_{\mathbb{C}^n}$. The symbol $\mathcal{M}_{k,d}$ denotes the Hilbert space of $k \times d$ linear operators over the complex field under the Hilbert-Schmidt inner product $\langle M, N\rangle := \mathrm{Tr}\,[M^\dagger N]$, and $\mathcal{M}_d := \mathcal{M}_{d,d}$. Let $\mathcal{U}(n)$ denote the set of $n \times n$ unitary matrices with complex entries. For a composite Hilbert space $\mathbb{C}^k \otimes \mathbb{C}^d$, the notation $\mathrm{Tr}_{\mathbb{C}^d}[\cdot]$ denotes the operation of taking partial trace i.e. tracing out the mentioned subsystem $\mathbb{C}^d$. We use $\mathrm{Tr}\,[\cdot]$ to denote the trace of the underlying operator. Fix standard bases for Hilbert spaces $A \cong \mathbb{C}^k$, $B \cong \mathbb{C}^d$. Let $|e_i\rangle^A$, $|e_i\rangle^B$ denote standard basis vectors of A, B respectively. Any vector $x \in A \otimes B$ can be written as $x = \sum_{i,j} \alpha_{ij} |e_i\rangle^A \otimes |e_j\rangle^B$. We use $\mathrm{op}_{d\to k}(x)$ to denote the operator $\sum_{i,j} \alpha_{ij} |e_i\rangle^A \otimes \langle e_j|^B$ in $\mathcal{M}_{k,d}$. Conversely, given an operator $M = \sum_{ij} m_{ij} |e_i\rangle^A \otimes \langle e_j|^B$ in $\mathcal{M}_{k,d}$, we let $\mathrm{vec}(M) := \sum_{ij} m_{ij} |e_i\rangle^A \otimes |e_j\rangle^B$ denote the vector in $\mathbb{C}^k \otimes \mathbb{C}^d$.

For Hermitian positive semidefinite operators M, we define M^α for any $\alpha > 0$ to be the unique Hermitian operator obtained by keeping the eigenbasis same and taking the αth power of the eigenvalues. We can define $\log M$ similarly. For $p > 1$, the notation $\|M\|_p$ denotes the Schatten p-norm of the matrix M, which is nothing but the ℓ_p-norm of the vector of its singular values. Alternatively, $\|M\|_p = (\mathrm{Tr}\,[(M^\dagger M)^{p/2}])^{1/p}$. Then $p = 2$ gives the Hilbert Schmidt norm aka the Frobenius norm which is nothing but $\|M\|_2 = \|\mathrm{vec}(M)\|_2$. Also, $p = \infty$ gives the operator norm aka spectral norm which is nothing but $\|M\|_\infty = \max_{v:\|v\|_2=1} \|Mv\|_2$.

Unless stated otherwise, the symbol ρ denotes a quantum state aka density matrix which is nothing but a Hermitian, positive semidefinite matrix with unit trace. A rank one density matrix is called a pure state. By the spectral theorem, any density matrix is a convex combination of pure states. The notation $\mathcal{D}(\mathbb{C}^d)$ denotes the convex set of all $d \times d$ density matrices. We use $|\cdot\rangle$ to denote a unit vector. By a slight abuse of notation, we shall often use a unit vector $|\psi\rangle$ to denote a pure state $|\psi\rangle\langle\psi|$. A linear mapping $\Phi : \mathcal{M}_m \to \mathcal{M}_d$ is called a superoperator. A superoperator is trace preserving if $\mathrm{Tr}\,\Phi(M) = \mathrm{Tr}\,M$ for all $M \in \mathcal{M}_m$. It is said to be positive if $\Phi(M)$ is positive semidefinite for all positive semidefinite M. Furthermore, Φ is said to be completely positive if $\Phi \otimes \mathbb{I}$ is a positive superoperator for identity superoperators $\mathbb{I}$ of all dimensions. Completely positive and trace preserving (CPTP) superoperators are referred to as quantum channels. Unless stated otherwise, Φ, Ψ are used to denote quantum channels.

A compact convex set $\mathcal{S}$ in $\mathbb{C}^n$ is called a convex body. The radius $r(\mathcal{S})$ of a convex body $\mathcal{S}$ is defined as

$$r(\mathcal{S}) := \min_{x\in\mathcal{S}} \max_{y\in\mathcal{S}} \|x - y\|_2.$$

Any point $x \in \mathcal{S}$ achieving the minimum above is said to be a centre of $\mathcal{S}$. The convex body $\mathcal{S}$ is said to be centrally symmetric iff for every $x \in \mathbb{C}^n$, $x \in \mathcal{S} \leftrightarrow -x \in \mathcal{S}$. The zero vector is a centre of a centrally symmetric convex body. A centrally symmetric convex body lying in $\mathbb{C}^n$ can be thought of as the unit sphere of a suitable notion of norm in $\mathbb{C}^n$. Conversely for any norm in $\mathbb{C}^n$, the unit sphere under the norm forms a centrally symmetric convex body.

2.1 Entropies and norms

▶ **Definition 1.** *The von Neumann entropy of a quantum state ρ is defined as*

$$S(\rho) := -\mathrm{Tr}\left[\rho \log \rho\right].$$

For all $p > 1$, the Rényi p-entropy of a quantum state ρ is defined as

$$S_p(\rho) := \frac{1}{1-p} \log \mathrm{Tr}\, \rho^p = -\frac{p}{p-1} \log \|\rho\|_p.$$

It turns out that $S(\rho) = \lim_{p \downarrow 1} S_p(\rho) =: S_1(\rho)$.
Also, it can be shown that for $p \geq 1$, $S_p(\cdot)$ is concave in its argument.

▶ **Definition 2.** *For $p \geq 1$, the minimum output Rényi p-entropy of a quantum channel Φ is defined as :*

$$S_p^{\min}(\Phi) := \min_{\rho \in \mathcal{D}(\mathbb{C}^m)} S_p(\Phi(\rho))$$

By an easy concavity argument it can be seen that above minimum is achieved on a pure state. Equivalently, to obtain $S_p^{\min}(\Phi)$ for $p > 1$ we must maximise $\|\Phi(\rho)\|_p$ for all input states ρ. This quantity is also known as the $1 \to p$ superoperator norm of superoperator $\Phi : \mathcal{M}_m \to \mathcal{M}_d$:

$$\|\Phi\|_{1 \to p} := \max_{M \in \mathcal{M}_m : \|M\|_1 = 1} \|\Phi(M)\|_p.$$

By an easy convexity argument it can be seen that the above maximum is achieved on a pure state i.e.

$$\|\Phi\|_{1 \to p} = \max_{x \in \mathbb{C}^m : \|x\|_2 = 1} \| |x\rangle\langle x| \|_p.$$

Thus, the additivity conjecture for minimal output p-Rényi p-entropy, $p > 1$, for quantum channels Φ and Ψ is equivalent to multiplicativity of $1 \to p$-norms of quantum channels viz. $\|\Phi \otimes \Psi\|_{1 \to p} \stackrel{?}{=} \|\Phi\|_{1 \to p} \cdot \|\Psi\|_{1 \to p}$. This equivalence will be used in Section B to give a counter example to additivity conjecture for all $p > 1$ where the Stinespring dilation of the quantum channel will be described from a unitary chosen uniformly at random from an approximate t-design. The equivalent result for Haar random unitaries was originally proved by Hayden and Winter [11].

We heavily use the one-one correspondence between quantum channels and subspaces of composite Hilbert spaces, originally proved by Aubrun, Szarek and Werner [4], in this paper. Let $\mathcal{W}$ be a subspace of $\mathbb{C}^k \otimes \mathbb{C}^d$ of dimension m. Identify $\mathcal{W}$ with $\mathbb{C}^m$ through an isometry $V : \mathbb{C}^m \to \mathbb{C}^k \otimes \mathbb{C}^d$ whose range is $\mathcal{W}$. Then, the corresponding quantum channel $\Phi_{\mathcal{W}} : \mathcal{M}_m \to \mathcal{M}_k$ is defined by $\Phi_{\mathcal{W}}(\rho) := \mathrm{Tr}_{\mathbb{C}^d}(V \rho V^\dagger)$. Using this equivalence and the fact that for $p > 1$ the $1 \to p$-superoperator norm is achieved on pure input states, we can write [4]

$$\|\Phi_{\mathcal{W}}\|_{1 \to p} = \max_{x \in \mathcal{W} : \|x\|_2 = 1} \|\mathrm{Tr}_{\mathbb{C}^d} |x\rangle\langle x|\|_p = \max_{x \in \mathcal{W} : \|x\|_2 = 1} \|\mathrm{op}_{d \to k}(x)\|_{2p}^2. \tag{1}$$

In an important paper, Shor [26] proved that several additivity conjectures for quantum channels were in fact equivalent to the additivity of minimum output von Neumann entropy of a quantum channel. More specifically, Shor showed that if there is a quantum channel Φ whose minimum output von Neumann entropy is subadditive, then there are quantum channels Ψ_1, Ψ_2 exhibiting superadditive classical Holevo capacity viz. $C(\Psi_1 \otimes \Psi_2) > C(\Psi_1) + C(\Psi_2)$. This equivalence was used as a starting point by Hastings [10] in his proof that there are channels with superadditive classical Holevo capacity. Aubrun, Szarek and Werner [3], as well as this paper also have the same starting point. For this, we need the following fact.

▶ **Fact 3** ([3, Lemma 2]). *Let a quantum channel $\Phi_{\mathcal{W}} : \mathcal{M}_m \to \mathcal{M}_k$ be described by a subspace $\mathcal{W} \leq \mathbb{C}^k \otimes \mathbb{C}^d$ of dimension m. Then,*

$$\begin{aligned} S_{\min}(\Phi_{\mathcal{W}}) &= \log k - k \cdot \max_{\rho \in \mathcal{D}(\mathbb{C}^m)} \|\Phi(\rho) - \frac{\mathbb{1}}{k}\|_2^2 \\ &= \log k - k \cdot \max_{x \in \mathcal{W}: \|x\|_2 = 1} \|(\mathrm{op}_{d \to k}(x))(\mathrm{op}_{d \to k}(x))^\dagger - \frac{\mathbb{1}}{k}\|_2^2. \end{aligned}$$

We will need the following result proved by Hayden and Winter [11] that upper bounds $S_p^{\min}(\Phi \otimes \bar{\Phi})$ where $\bar{\Phi}$ denotes the CPTP superoperator obtained by taking complex conjugate of the CPTP superoperator Φ.

▶ **Fact 4.** *Let $V : \mathbb{C}^m \to \mathbb{C}^k \otimes \mathbb{C}^d$ be an isometry describing the quantum channel $\Phi : \rho \mapsto \mathrm{Tr}_{\mathbb{C}^d}[V \rho V^\dagger]$. Let $|\phi\rangle$ denote the maximally entangled state in $\mathbb{C}^m \otimes \mathbb{C}^m$. Suppose $m \leq d$. Then $(\Phi \otimes \bar{\Phi})(|\phi\rangle\langle\phi|)$ has a singular value not less than $\frac{m}{kd}$. Hence for all $p > 1$,*

$$\|\Phi \otimes \bar{\Phi}\|_{1 \to p} \geq \|\Phi \otimes \bar{\Phi}\|_{1 \to \infty} \geq \frac{m}{kd}.$$

Moreover,

$$S_{\min}(\Phi \otimes \bar{\Phi}) \leq 2 \log k - \frac{m}{kd} \log k + O\left(\frac{m}{kd} \log \frac{d}{m} + \frac{1}{k}\right).$$

2.2 Concentration results for Lipschitz functions

We now state some basic definitions and facts from geometric functional analysis that will be used in the proof of our main result.

▶ **Definition 5.** *A function $f : X \to \mathbb{C}$ defined over a metric space X is said to be L-Lipschitz if $\forall x, y \in X$ it satisfies the following inequality:*

$$|f(x) - f(y)| \leq L \cdot d(x, y).$$

▶ **Definition 6.** *Let X be a compact metric space. An ϵ-net $\mathcal{N}$ of X is a finite set of points such that for any point $x \in X$, there is a point $x' \in \mathcal{N}$ such that $d(x, x') \leq \epsilon$.*

Note that compactness guarantees that finite sized ϵ-nets exist for all $\epsilon > 0$.

We will need the following definition and fact from [3].

▶ **Definition 7.** *A function $f : X \to \mathbb{C}$ defined over a normed linear space X is said to be circled if $f(e^{i\theta} x) = f(x)$ for all $\theta \in \mathbb{R}$ and $x \in X$.*

▶ **Fact 8.** *Let $f : X \to \mathbb{R}$ be a function defined on a metric space X. Suppose there exists a subset $Y \subseteq X$ such that f restricted to Y is L-Lipschitz. Then there is a function $\hat{f} : X \to \mathbb{R}$ that is L-Lipschitz on all of X satisfying $\hat{f}(y) = f(y)$ for all $y \in Y$. If X is a normed linear space over real or complex numbers and f is circled then the extension $\hat{f}$ is also circled.*

Proof Sketch. Define $\hat{f}(x) := \inf_{y \in Y}[f(y) + Ld(x,y)]$. ◀

In this paper, we endow $\mathbb{C}^n$ with the ℓ_2-metric and $\mathbb{U}(n)$ with the Schatten ℓ_2-metric aka Frobenius metric. The following fact gives a reasonably tight upper bound on the size of an ϵ-net of $\mathbb{S}_{\mathbb{C}^n}$.

▶ **Fact 9** ([28, Corollary 4.2.13]). *Let $\epsilon > 0$. There exists an ϵ-net of $\mathbb{S}_{\mathbb{C}^n}$ of size less than $(\frac{3}{\epsilon})^{2n}$.*

A fundamental result about concentration of Lipschitz functions defined on the unit sphere or the unitary group, known as Levy's lemma, lies at the heart of all proofs of Dvoretzky-type theorems via the probabilistic method. We now state the version of Levy's lemma that will be used in this paper.

▶ **Fact 10** (Levy's lemma, [2, Corollary 4.4.28]). *Consider the Haar probability measure on $\mathbb{S}_{\mathbb{C}^n}$. Let $f : \mathbb{S}_{\mathbb{C}^n} \to \mathbb{C}$ be an L-Lipshitz function. Let $\mu := \mathbb{E}_x[f(x)]$ and $\lambda > 0$. Then*

$$\Pr_x(|f(x) - \mu| \geq \lambda) \leq 2\exp(-\frac{n\lambda^2}{4L^2}).$$

An elementary proof of the above fact, without explicitly calculated constants, can be found in [28, Theorem 5.1.4].

For our work, we need a measure concentration inequality like Levy's lemma for difference of function values on two distinct arbitrary points which is sensitive to the distance between those points. Such an inequality is stated in the following fact.

▶ **Fact 11** ([3, Lemma 9]). *Let $f : \mathbb{S}_{\mathbb{C}^n} \to \mathbb{C}$ be a circled L-Lipschitz function. Consider the Haar probability measure on $\mathbb{U}(n)$. Then for any $x, y \in \mathbb{S}_{\mathbb{C}^n}$, $x \neq y$ and for any $\lambda > 0$,*

$$\Pr_U[|f(Ux) - f(Uy)| > \lambda] \leq 2\exp(-\frac{\lambda^2 n}{8L^2\|x-y\|_2^2}).$$

The derandomisation in our paper is carried out by replacing the Stinespring dilation unitary of a quantum channel, which is chosen from the Haar measure in [3], with a unitary chosen uniformly at random from a finite cardinality approximate unitary t-design for a suitable value of t. The next few statements lead us to the definition of an approximate unitary t-design.

▶ **Definition 12** ([16, Definition 2.2]). *A monomial in the entries of a matrix U is of degree (r, s) if it contains r conjugated elements and s unconjugated elements. The evaluation of monomial M at the entries of a matrix U is denoted by $M(U)$. We call a monomial balanced if $r = s$, and say that it has degree t if it is of degree (t, t). A polynomial is said to be balanced of degree t if it is a sum of balanced monomials of degree at most t.*

▶ **Definition 13** ([16, Definition 2.3]). *A probability distribution ν supported on a finite set of $d \times d$ unitary matrices is said to be an exact unitary t-design if for all balanced monomials M of degree at most t, $\mathbb{E}_{U\sim\nu}[M(U)] = \mathbb{E}_{U\sim\text{Haar}}[M(U)]$.*

▶ **Definition 14** ([16, Definition 2.6]). *A probability distribution ν supported on a finite set of $d \times d$ unitary matrices is said to be an ϵ-approximate unitary t-design if for all balanced monomials M of degree at most t*

$$|\mathbb{E}_{U\sim\nu}(M(U)) - \mathbb{E}_{U\sim\text{Haar}}(M(U))| \leq \frac{\epsilon}{d^t}.$$

We will need the following fact.

▶ **Fact 15** ([16, Lemma 3.4]). *Let $Y : \mathbb{U}(n) \to \mathbb{C}$ be a balanced polynomial of degree a in the entries of the unitary matrix U that is provided as input. Let $\alpha(Y)$ denote the sum of absolute values of the coefficients of Y. Let r, t be positive integers satisfying $2ar < t$. Let ν be an ϵ-approximate unitary t-design. Then*

$$\mathbb{E}_{U\sim\nu}[|Y_U|^{2r}] \leq \mathbb{E}_{U\sim\text{Haar}}[|Y_U|^{2r}] + \frac{\epsilon\alpha(Y)^{2r}}{n^t}.$$

3 Sharp Dvoretzky-like theorems via stratified analysis

In this section, we prove our main technical results viz. sharp Dvoretzky-like theorems for Haar measure as well as approximate t-designs using stratified analysis. We start by stating the following three lemmas, with their proofs in Appendix A which are 'baby stratified' analogues of Fact 11 for Haar measure and approximate unitary t-designs.

▶ **Lemma 16.** *Let $Y : \mathbb{S}_{\mathbb{C}^n} \to \mathbb{R}$ be a circled function with global Lipschitz constant L_1. Suppose that there exists a subset $\Omega \subseteq \mathbb{S}_{\mathbb{C}^n}$ such that Y restricted to Ω has a smaller Lipschitz constant L_2. Let $x, y \in \mathbb{S}_{\mathbb{C}^n}$. Let $Y_x := Y(Ux)$, $Y_y := Y(Uy)$ be two correlated random variables, under the choice of a Haar random unitary U. Let $\lambda > 0$. Then*

$$\Pr_{U\sim\text{Haar}}[|Y_x - Y_y| > \lambda] \leq 2\exp(-\frac{n\lambda^2}{8L_2^2\|x-y\|_2^2}) + 2\Pr_{z\sim\text{Haar}}[z \in \Omega^c].$$

▶ **Lemma 17.** *Let $Y : \mathbb{S}_{\mathbb{C}^n} \to \mathbb{R}$ be a circled function with global Lipschitz constant L_1. Suppose that there exists a subset $\Omega \subseteq \mathbb{S}_{\mathbb{C}^n}$ such that Y restricted to Ω has a smaller Lipschitz constant L_2. Let $Z : \mathbb{S}_{\mathbb{C}^n} \to \mathbb{R}$ be a balanced polynomial of degree a in entries of the vector $x \in \mathbb{C}^n$ that is provided as input. Let $\delta > 0$. Let $f : \mathbb{R} \to \mathbb{R}$ be a non-decreasing function. Suppose that $|Z(x) - f(Y(x))| < \delta$ for all $x \in \mathbb{S}_{\mathbb{C}^n}$. Let $x, y \in \mathbb{S}_{\mathbb{C}^n}$. Let $Z_x := Z(Ux)$, $Z_y := Z(Uy)$ be two correlated random variables, under the choice of a unitary U chosen uniformly at random from an ϵ-approximate unitary t-design ν. Let r be a positive integer satisfying $2ar \leq t$. Let $0 < \epsilon < \frac{n^{t-r}(4rL_2^2\|x-y\|_2^2)^r}{\alpha(Z)^{2r}}$. Then*

$$\mathbb{E}_{U\sim\nu}[|Z_x - Z_y|^{2r}] \leq 3\left(\frac{4rL_2^2\|x-y\|_2^2}{n}\right)^r + 2\Pr_{z\sim\text{Haar}}[z \in \Omega^c] \cdot (L_1^2\|x-y\|_2^2)^r.$$

▶ **Lemma 18.** *Let $Y : \mathbb{S}_{\mathbb{C}^n} \to \mathbb{R}$ be a balanced polynomial of degree a in entries of the vector $x \in \mathbb{C}^n$ that is provided as input. Let $\alpha(Y)$ denote the sum of absolute values of the coefficients of Y. Suppose Y has global Lipschitz constant L_1. Suppose that there exists a subset $\Omega \subseteq \mathbb{S}_{\mathbb{C}^n}$ such that Y restricted to Ω has a smaller Lipschitz constant L_2. Let $x, y \in \mathbb{S}_{\mathbb{C}^n}$. Let $Y_x := Y(Ux)$, $Y_y := Y(Uy)$ be two correlated random variables, under the choice of a unitary U chosen uniformly at random from an ϵ-approximate unitary t-design ν. Let r be a positive integer satisfying $2ar \leq t$. Let $0 < \epsilon < \frac{n^{t-r}(4rL_2^2\|x-y\|_2^2)^r}{\alpha(Y)^{2r}}$. Then*

$$\mathbb{E}_{U\sim\nu}[|Y_x - Y_y|^{2r}] \leq 3\left(\frac{4rL_2^2\|x-y\|_2^2}{n}\right)^r + 2\Pr_{z\sim\text{Haar}}[z \in \Omega^c] \cdot (L_1^2\|x-y\|_2^2)^r.$$

We also need a so-called *chaining inequality* for probability similar to Dudley's inequality in geometric functional analysis [3, 21]. The original Dudley's inequality bounds the expectation of the supremum, over pairs of correlated random variables, of the difference between them in terms of an integral, over η, of a certain function of the size of an η-net of $\mathbb{S}_{\mathbb{C}^n}$. Our

chaining lemma differs from it in two important respects. First, instead of the expectation it bounds a tail probability of the supremum, over pairs of correlated random variables, of the difference between them. Second, it replaces the integral by a finite summation over η-nets of $\mathbb{S}_{\mathbb{C}^n}$ with geometrically decreasing η. Despite the fancy name, our chaining lemma is a simple consequence of the union bound of probabilities. Nevertheless, it is crucial to proving our main result as it allows us to efficiently invoke powerful measure concentration results in order to bound the variation of a Lipschitz function on subspaces of $\mathbb{C}^n$.

▶ **Lemma 19** (Chaining). *Let $\{X_s\}_{s\in\mathcal{S}}$ be a family of correlated complex valued random variables indexed by elements of a compact metric space $\mathcal{S}$. Let $\lambda, L_1 > 0$. The family is said to be L_1-Lipschitz if for all $s,t \in \mathcal{S}$, $|X_s - X_t| \leq L_1 d(s,t)$ for all points of the sample space. Define i_0 to be the unique integer such that the radius of $\mathcal{S}$ lies in the interval $(2^{-i_0-1}, 2^{-i_0}]$. Define $i_1 := \max\{i_0, \lceil \log \frac{2L_1}{\lambda} \rceil\}$. Let $p : \mathbb{Z} \to \mathbb{R}_+$ be a non-decreasing function. Suppose the infinite series $\sum_{i>i_0} \frac{\sqrt{|i|p(i)}}{2^i}$ is convergent with value C. Then,*

$$\Pr[\sup_{s,t\in\mathcal{S}} |X_s - X_t| > \lambda] \leq \sum_{i=i_0+1}^{i_1+1} \sum_{(u,u')\in\mathcal{N}_{i-1}\times\mathcal{N}_i : d(u,u')<2^{-i+2}} \Pr[|X_u - X_{u'}| > \frac{\lambda\sqrt{|i|p(i)}}{4C\cdot 2^i}],$$

for a sequence of 2^{-i}-nets $\mathcal{N}_i$, $i_0 \leq i \leq i_1$, $|\mathcal{N}_{i_0}| = 1$, of $\mathcal{S}$.

Proof. For every $i \in \mathbb{Z}$, let $\mathcal{N}_i$ be a 2^{-i}-net of $\mathcal{S}$. Let i_0 be such that radius of $\mathcal{S}$ lies in $(2^{-(i_0+1)}, 2^{-i_0}]$. The net $\mathcal{N}_{i_0}$ consists of a single element, say s_0. For every $s \in \mathcal{S}$ and $i \in \mathbb{Z}$, let $\pi_i(s)$ be an element of $\mathcal{N}_i$ satisfying $d(s, \pi_i(s)) \leq 2^{-i}$. We have the following chaining equation for every $s \in \mathcal{S}$:

$$X_s = X_{s_0} + \left(\sum_{i=i_0}^{i_i} (X_{\pi_{i+1}(s)} - X_{\pi_i(s)})\right) + (X_s - X_{\pi_{i_1+1}(s)}).$$

Lipschitz property of the family implies that

$$\begin{aligned}
\sup_{s,t\in\mathcal{S}} |X_s - X_t| &\leq 2\sum_{i=i_0}^{i_1} \sup_{s\in\mathcal{S}} |X_{\pi_{i+1}(s)} - X_{\pi_i(s)}| + L_1 2^{-i_1} \\
&\leq 2\sum_{i=i_0}^{i_1} \sup_{(u,u')\in\mathcal{N}_i\times\mathcal{N}_{i+1} : d(u,u')<2^{-i+1}} |X_u - X_{u'}| + L_1 2^{-i_1} \\
&\leq 2\sum_{i=i_0+1}^{i_1+1} \sup_{(u,u')\in\mathcal{N}_{i-1}\times\mathcal{N}_i : d(u,u')<2^{-i+2}} |X_u - X_{u'}| + \frac{\lambda}{2}.
\end{aligned}$$

Now if $\sup_{s,t\in\mathcal{S}} |X_s - X_t| > \lambda$, there must exist an i, $i_0 + 1 \leq i \leq i_1 + 1$ such that

$$\sup_{(u,u')\in\mathcal{N}_{i-1}\times\mathcal{N}_i : d(u,u')<2^{-i+2}} |X_u - X_{u'}| > \frac{\lambda\sqrt{|i|p(i)}}{4C\cdot 2^i}.$$

Applying the union bound on probability leads us to the conclusion of the lemma. ◀

We now prove our sharp Dvoretzky-like theorem for subspaces chosen from the Haar measure using stratified analysis.

▶ **Theorem 20.** *Let $p : \mathbb{N} \to \mathbb{R}_+$ be a non-decreasing function. Suppose the infinite series $\sum_{i>0} \frac{\sqrt{ip(i)}}{2^i}$ is convergent with value C. Let $f : \mathbb{S}_{\mathbb{C}^n} \to \mathbb{R}$ have global Lipschitz constant L_1. Let $L_2, c_1, c_2, c_3, \lambda > 0$. Define $m := \lceil \frac{c_1 n \lambda^2}{L_2^2} \rceil$. Suppose there is an increasing sequence of subsets $\Omega_1 \subseteq \Omega_2 \subseteq \cdots$ of $\mathbb{S}_{\mathbb{C}^n}$ such that with probability at least $1 - c_2 e^{-c_3 mi}$, a Haar random subspace of dimension m lies in Ω_i and f restricted to Ω_i has Lipschitz constant $L_2\sqrt{p(i)}$. Then there exists a constant c depending on c_3, C, $0 < c < 1$, such that for $m' := cm$ with probability at least $1 - (c_2 + 1)2^{-m'}$, a subspace W of dimension m' chosen with respect to Haar measure satisfies the property that $|f(w) - \mu| < \lambda$ for all points $w \in W \cap \mathbb{S}_{\mathbb{C}^n}$.*

Proof. In this proof $\mathbb{S}_{\mathbb{C}^n}$ denotes the unit ℓ_2-length sphere in $\mathbb{C}^n$ together with the origin point 0. The radius of $\mathbb{S}_{\mathbb{C}^n}$ is one which makes $i_0 = 0$ in Lemma 19. Consider a canonical embedding of $\mathbb{S}_{\mathbb{C}^{m'}}$ into $\mathbb{S}_{\mathbb{C}^m}$ and further into $\mathbb{S}_{\mathbb{C}^n}$. Define

$$B_i := \{U \in \mathbb{U}(n) : \forall z \in \mathbb{S}_{\mathbb{C}^m}, Uz \in \Omega_i\}.$$

For $s \in \mathbb{S}_{\mathbb{C}^{m'}}$, define the random variable $Y_s := f(Us) - \mu$, where the randomness arises solely from the choice of $U \in \mathbb{U}(n)$. Then $\Pr_{U\sim\text{Haar}}[B_i] \geq 1 - c_2 e^{-c_3 mi}$.

Let $i_1 := \lceil \log \frac{2L_1}{\lambda} \rceil$. Let $\mathcal{N}_i$, $i = 0, 1, \ldots, i_1$ be a sequence of 2^{-i}-nets in $\mathbb{S}_{\mathbb{C}^{m'}}$ of minimum cardinality, where $\mathcal{N}_0 := \{0\}$ and $Y_0 := 0$. We can take $|\mathcal{N}_i| \overset{a}{\leq} 2^{2(i+2)m'}$ by Fact 9. By Lemma 19

$$\Pr_{U\sim\text{Haar}}[\sup_{s,t\in\mathbb{S}_{\mathbb{C}^{m'}}} |Y_s - Y_t| > \lambda] \leq 2\sum_{i=1}^{i_1+1} \sum_{(u,u')\in\mathcal{N}_{i-1}\times\mathcal{N}_i : \|u-u'\|_2 < 2^{-i+2}} \Pr_{U\sim\text{Haar}}[|Y_u - Y_{u'}| > \frac{\lambda\sqrt{ip(i)}}{4C\cdot 2^i}].$$

Applying Lemma 16 to the set B_i gives, for u, u' satisfying $\|u - u'\|_2 < 2^{-i+2}$,

$$\begin{aligned}
&\Pr_{U\sim\text{Haar}}[|Y_u - Y_{u'}| > \frac{\lambda\sqrt{ip(i)}}{4C\cdot 2^i}] \\
&\leq\ 2\exp\left(-\frac{n\lambda^2 ip(i)}{2^7 C^2 2^{2i} L_2^2 p(i)\|u-u'\|_2^2}\right) + 2\Pr_{z\sim\text{Haar}}[z \in \Omega_i^c] \\
&\leq\ 2\exp\left(-\frac{ni\lambda^2}{2^9 C^2 L_2^2}\right) + 2\Pr_{z\sim\text{Haar}}[z \in \Omega_i^c] \\
&\leq\ 2\exp\left(-\frac{im}{2^9 C^2}\right) + 2c_2\exp(-c_3 mi) \ \leq\ 2(c_2+1)\exp(-c_4 mi),
\end{aligned}$$

for a constant c_4 depending only on C and c_3.

This gives us

$$\begin{aligned}
&\Pr_{U\sim\text{Haar}}[\sup_{s,t\in\mathbb{S}_{\mathbb{C}^{m'}}} |Y_s - Y_t| > \lambda] \\
&\leq\ 4(c_2+1)\sum_{i=1}^{i_1+1} \sum_{(u,u')\in\mathcal{N}_{i-1}\times\mathcal{N}_i : \|u-u'\|_2 < 2^{-i+2}} e^{-c_4 mi} \ \leq\ 4(c_2+1)\sum_{i=1}^{i_1+1} |\mathcal{N}_{i-1}|\cdot|\mathcal{N}_i|\cdot e^{-c_4 mi} \\
&\leq\ 4(c_2+1)\sum_{i=1}^{i_1+1} 2^{4m'(i+2)} e^{-c_4 mi} \ \leq\ (c_2+1)2^{-m'},
\end{aligned}$$

where the third inequality follows from (a) and the fourth inequality follows from the definition $m' := cm$ for an appropriate choice of c depending only on c_4. In other words, c depends only on C and c_3.

Taking $t = 0$, we see that with probability at least $1 - (c_2 + 1)2^{-m'}$ over the choice of a Haar random unitary, we have that for all $s \in \mathbb{S}_{\mathbb{C}^{m'}}$, $|Y_s| \leq \lambda$. This completes the proof of the theorem. ◀

Remark

The sets Ω_i and the Lipschitz constants $L_2\sqrt{p(i)}$ for $1 \leq i \leq \lceil \log \frac{2L_1}{\lambda} \rceil + 1$ formalise the idea of stratified analysis mentioned intuitively in the introduction. As i increases the relevant Lipschitz constant increases. So we need a finer net i.e. a 2^{-i}-net for the ith layer Ω_i in order to control the variation of f for subspaces lying inside Ω_i. With exponentially high probability, we thus get a Haar random subspace of dimension m', slightly smaller than m, where f is almost constant. Note that the definition of m involves only the smallest local Lipschitz constant L_2. Thus the dimension of the space m' that we obtain is larger than what would be obtained by a naive analysis which would be constrained by the global Lipschitz constant L_1. Moreover, a naive analysis would not give exponentially high probability, just an arbitrary constant close one. These two properties underscore the power of our stratified analysis. However, applying the stratified analysis to a concrete function is not always straightforward. We need to define the layers $\Omega_1, \Omega_2, \ldots$, properly and show separately that Haar random subspaces of dimension m lie in Ω_i with probability $1 - c_2 e^{-c_3 m i}$. But for several interesting functions this can be done without much difficulty. This will become clearer in Section 4 where we will show how to recover Aubrun, Szarek and Werner's result for the Haar measure directly from Theorem 20, without having to apply a Dvoretzky-style theorem twice in a messy fashion as in the original paper [3]. Moreover, we get success probability exponentially close to one unlike Aubrun, Szarek and Werner who could get only a constant close to one. Furthermore, our methods extend to approximate t-designs and allows us to prove exponentially close to one probability even for that setting.

We now prove our sharp Dvoretzky-like theorem for subspaces chosen from approximate t-designs using stratified analysis.

▶ **Theorem 21.** *Let $p : \mathbb{N} \to \mathbb{R}_+$ be a non-decreasing function. Suppose the infinite series $\sum_{i>0} \frac{\sqrt{ip(i)}}{2^i}$ is convergent with value C. Let $f : \mathbb{S}_{\mathbb{C}^n} \to \mathbb{R}$ be a balanced degree 'a' polynomial with global Lipschitz constant L_1. Let $0 \leq L_2 \leq 1$, $c_1, c_2, c_3, \lambda > 0$. Define $m := \lceil \frac{c_1 n \lambda^2}{L_2^2} \rceil$. Suppose there is an increasing sequence of subsets $\Omega_1 \subseteq \Omega_2 \subseteq \cdots$ of $\mathbb{S}_{\mathbb{C}^n}$ such that with probability at least $1 - c_2 e^{-c_3 m i}$, a Haar random subspace of dimension m lies in Ω_i and f restricted to Ω_i has Lipschitz constant $L_2\sqrt{p(i)}$. Suppose*

$$0 < \epsilon < \left(\frac{\lambda}{4L_1}\right)^{2m} \cdot \frac{n^{(2a-1)m}(L_2^2 p(1))^m}{\max\{\alpha(f)^{2m}, 1\}}.$$

Then there exists a constant c depending on c_1, c_3, C, $p(1)$, $0 < c < 1$ such that for

$$m' := cm \frac{\log \log \frac{C^2 L_1^2}{\lambda^2 p(1)}}{\lceil \log \frac{C^2 L_1^2}{\lambda^2 p(1)} \rceil},$$

with probability at least $1 - (c_2 + 1)2^{-m'}$, a subspace W of dimension m' chosen under an ϵ-approximate $(2am)$-design ν satisfies the property that $|f(w) - \mu| < \lambda$ for all points $w \in W \cap \mathbb{S}_{\mathbb{C}^n}$.

Proof. In this proof $\mathbb{S}_{\mathbb{C}^n}$ denotes the unit ℓ_2-length sphere in $\mathbb{C}^n$ together with the origin point 0. The radius of $\mathbb{S}_{\mathbb{C}^n}$ is one which makes $i_0 = 0$ in Lemma 18. Consider a canonical embedding of $\mathbb{S}_{\mathbb{C}^{m'}}$ into $\mathbb{S}_{\mathbb{C}^m}$ and further into $\mathbb{S}_{\mathbb{C}^n}$. Define

$$B_i := \{U \in \mathbb{U}(n) : \forall z \in \mathbb{S}_{\mathbb{C}^m}, Uz \in \Omega_i\}.$$

For $s \in \mathbb{S}_{\mathbb{C}^{m'}}$, define the random variable $Y_s := f(Us) - \mu$, where the randomness arises solely from the choice of $U \in \mathbb{U}(n)$. Then $\Pr_{U\sim\mathrm{Haar}}[B_i] \geq 1 - c_2 e^{-c_3 mi}$.

Let $i_1 := \lceil \log \frac{2L_1}{\lambda} \rceil$. Let $\mathcal{N}_i$, $i = 0, 1, \ldots, i_1$ be a sequence of 2^{-i}-nets in $\mathbb{S}_{\mathbb{C}^{m'}}$ of minimum cardinality, where $\mathcal{N}_0 := \{0\}$ and $Y_0 := 0$. We can take $|\mathcal{N}_i| \overset{a}{\leq} 2^{2(i+2)m'}$ by Fact 9. By Lemma 19

$$\begin{aligned}
&\Pr_{U\sim\nu}[\sup_{s,t\in\mathbb{S}_{\mathbb{C}^{m'}}} |Y_s - Y_t| > \lambda] \leq \\
&2\sum_{i=1}^{i_1+1} \sum_{(u,u')\in\mathcal{N}_{i-1}\times\mathcal{N}_i : \|u-u'\|_2 < 2^{-i+2}} \Pr_{U\sim\nu}[|Y_u - Y_{u'}| > \frac{\lambda\sqrt{ip(i)}}{4C\cdot 2^i}]. \qquad (2)
\end{aligned}$$

Let r be a positive integer such that $r(i_1 + 1) < m$. Applying Lemma 18 to the set B_i gives, for u, u' satisfying $\|u - u'\|_2 < 2^{-i+2}$,

$$\begin{aligned}
&\Pr_{U\sim\nu}[|Y_u - Y_{u'}| > \frac{\lambda\sqrt{ip(i)}}{4C\cdot 2^i}] \\
&= \Pr_{U\sim\nu}[|Y_u - Y_{u'}|^{2ri} > \left(\frac{\lambda^2 ip(i)}{2^4 C^2 2^{2i}}\right)^{ri}] \leq \left(\frac{2^{2i+4}C^2}{\lambda^2 ip(i)}\right)^{ri} \mathbb{E}_{U\sim\nu}[|Y_u - Y_{u'}|^{2ri}] \\
&\leq 3\left(\frac{2^{2i+4}C^2}{\lambda^2 ip(i)}\right)^{ri} \left(\left(\frac{4riL_2^2 p(i)\|u-u'\|_2^2}{n}\right)^{ri} + c_2 e^{-c_3 mi}\cdot (L_1^2\|u-u'\|_2^2)^{ri}\right) \\
&\leq 3\left(\frac{2^{2i+6}C^2 r L_2^2\|u-u'\|_2^2}{n\lambda^2}\right)^{ri} + 3c_2 e^{-c_3 mi}\left(\frac{2^{2i+4}C^2L_1^2\|u-u'\|_2^2}{\lambda^2 ip(i)}\right)^{ri} \\
&\leq \underbrace{3\left(\frac{2^{10}C^2 r L_2^2}{n\lambda^2}\right)^{ri}}_{=:\mathrm{I}} + \underbrace{3c_2 e^{-c_3 mi}\left(\frac{2^8 C^2 L_1^2}{\lambda^2 p(1)}\right)^{ri}}_{=:\mathrm{II}}.
\end{aligned}$$

We now analyse the two terms in the above expression. Take

$$r := \frac{c_4 n\lambda^2}{2^{10}C^2L_2^2}\cdot\frac{1}{\lceil\log\frac{2^8C^2L_1^2}{\lambda^2 p(1)}\rceil}$$

for a constant c_4, $0 < c_4 < 1$, c_4 depending only on C, c_1, c_3, $p(1)$ chosen to be small enough so that $r(i_1 + 1) < m$ and $\frac{c_4 n\lambda^2}{2^{10}C^2L_2^2} \leq \frac{c_3 m}{2}$. Substitute r back in I and II to get

$$\mathrm{I} \leq 3\cdot 2^{-ri\log\log\frac{2^8C^2L_1^2}{\lambda^2 p(1)}}, \quad \mathrm{II} \leq 3c_2 e^{-c_3 mi} 2^{\frac{c_3 mi}{2}} < 3c_2 e^{-c_3 mi/2}.$$

We choose

$$m'' := r\log\log\frac{2^8C^2L_1^2}{\lambda^2 p(1)} < \frac{c_3 m}{2}.$$

This gives us

$$\mathrm{I} \leq 3\cdot 2^{-m''i}, \quad \mathrm{II} \leq 3c_2 e^{-m''i}.$$

Thus, we have shown that

$$\Pr_{U\sim\nu}[|Y_u - Y_{u'}| > \frac{\lambda\sqrt{p(i)}}{4C\cdot 2^i}] \leq 3(c_2+1)2^{-m''i}.$$

Substituting above in Equation 2, we get

$$\begin{aligned}
&\Pr_{U\sim\nu}[\sup_{s,t\in\mathbb{S}_{\mathbb{C}^{m'}}} |Y_s - Y_t| > \lambda] \\
&\leq \quad 2\sum_{i=1}^{i_1+1} \sum_{u,u'\in\mathcal{N}_{i-1}\times\mathcal{N}_i:\|u-u'\|<2^{-i+2}} 3(c_2+1)2^{-m''i} \\
&\leq \quad 6(c_2+1)\sum_{i=1}^{i_1+1} |\mathcal{N}_{i-1}|\cdot|\mathcal{N}_i|\cdot 2^{-m''i} \;\leq\; 6(c_2+1)\sum_{i=1}^{i_1+1} 2^{4m'(i+2)}2^{-m''i} \;\leq\; (c_2+1)2^{-m'},
\end{aligned}$$

if m' is chosen as indicated above for a small enough constant c, $0 < c < 1$, c depending only on c_4, c_1, C i.e. c depending only on C, c_1, c_3, $p(1)$.

Taking $t = 0$, we see that with probability at least $1 - (c_2+1)2^{-m'}$ over the choice of a uniformly random unitary from the approximate $(2am)$-design, we have that for all $s \in \mathbb{S}_{\mathbb{C}^{m'}}$, $|Y_s| \leq \lambda$. This completes the proof of the theorem. ◀

4 Strict subadditivity of minimum output von Neumann entropy for approximate t-designs

We first apply Theorem 20 in order to directly recover Aubrun, Szarek and Werner's result [3] that channels with Haar random unitary Stinespring dilations exhibit strict subadditivity of minimum output von Neumann entropy. In fact, we go beyond their result in the sense that we obtain exponentially high probability close to one as opposed to constant probability. After this warmup, we apply Theorem 21 in order to show that channels with approximate $n^{2/3}$-design unitary Stinespring dilations exhibit strict subadditivity of minimum output von Neumann entropy with exponentially high probability close to one.

Let k be a positive integer. Consider the sphere $\mathbb{S}_{\mathbb{C}^{k^3}}$. Define the $k \times k^2$ matrix M to be the rearrangment of a k^3-tuple from $\mathbb{S}_{\mathbb{C}^{k^3}}$. Note that the ℓ_2-norm on $\mathbb{C}^{k^3}$ is the same as the Frobenius norm on $\mathbb{C}^{k\times k^2}$.

In Step I, we define the function $f : \mathbb{S}_{\mathbb{C}^{k^3}} \to \mathbb{R}$ as $f(M) := \|M\|_\infty$. The function f has global Lipschitz constant $L_1 = 1$ since

$$|f(M) - f(N)| \leq \|M - N\|_\infty \leq \|M - N\|_2.$$

For large enough k the mean μ of f, under the Haar measure, is less than $2k^{-1/2}$ [3, Corollary 7]. We use the notation of Theorem 20. Define $L_2 := 1$, $p(i) := 1$ for all $i \in \mathbb{N}$. Then $C < 2$. Define the layers $\Omega_1, \Omega_2, \ldots$, to be all of $\mathbb{S}_{\mathbb{C}^{k^3}}$. Let j, $4 \leq j \leq k$ be a positive integer. Let $\lambda_j := \sqrt{\frac{j}{k}}$. Define $c_1 := 1$, $m = k^2$, $c_2 := 0$, $c_3 := 1$. Trivially, a Haar random subspace of dimension mj lies in Ω_i with probability at least $1 - c_2 e^{-c_3 mji}$. Theorem 20 tells us that there is a universal constant $\hat{c}_1$ such that for $m' := \hat{c}_1 k^2$, with probability at least $1 - 2^{-m'j}$, a Haar random subspace W of dimension $m'j$ satisfies

$$\|M\|_\infty < \frac{2}{\sqrt{k}} + \sqrt{\frac{j}{k}} < 2\sqrt{\frac{j}{k}}$$

for all $M \in W$.

In Step II, we define the function $f : \mathbb{S}_{\mathbb{C}^{k^3}} \to \mathbb{R}$ as $f(M) := \|MM^\dagger - \frac{\mathbb{1}}{k}\|_2$. The function f has global Lipschitz constant $L_1 = 2$ since

$$\begin{aligned} |f(M) - f(N)| &\leq \|MM^\dagger - NN^\dagger\|_2 \leq \|MM^\dagger - MN^\dagger\|_2 + \|MN^\dagger - NN^\dagger\|_2 \\ &\leq \|M\|_\infty \|M^\dagger - N^\dagger\|_2 + \|N^\dagger\|_\infty \|M - N\|_2 \\ &= (\|M\|_\infty + \|N\|_\infty)\|M - N\|_2 \leq 2\|M - N\|_2. \end{aligned}$$

The mean μ of f, under the Haar measure, is less than $c_0 k^{-1}$ for a universal constant c_0 [3, Corollary 7]. We use the notation of Theorem 20. Let j, $c_0 < j \leq k$ be a positive integer. Define $L_2 := 4\sqrt{\frac{j}{k}}$, $p(i) := i + 3$ for all $i \in \mathbb{N}$. Then $C \leq 4$. Define the layers $\Omega_1, \Omega_2, \ldots,$ to be the subsets

$$\Omega_i := \left\{ M \in \mathbb{S}_{\mathbb{C}^{k^3}} : \|M\|_\infty \leq 2\sqrt{\frac{j(i+3)}{k}} \right\}.$$

It is easy to see that f restricted to Ω_i has local Lipschitz constant at most $L_2\sqrt{p(i)}$. Let $\lambda := \frac{j}{k}$. Define $c_1 := 16\hat{c}_1$, $m = \hat{c}_1 j k^2$, $c_2 := 1$, $c_3 := \ln 2$. By the previous paragraph, a Haar random subspace of dimension $m(i+3)$ lies in Ω_i with probability at least $1 - c_2 e^{-c_3 m(i+3)} \geq 1 - c_2 e^{-c_3 mi}$. Theorem 20 tells us that there is a universal constant $\hat{c}_2$ such that for $m' := \hat{c}_2 k^2$, with probability at least $1 - 2^{-m'j}$, a Haar random subspace W of dimension $m'j$ satisfies

$$f(M) = \|MM^\dagger - \frac{\mathbb{1}}{k}\|_2 < \frac{c_0}{k} + \frac{j}{k} < \frac{2j}{k}$$

for all $M \in W$. Setting $j = 1$ allows us to recover Aubrun, Szarek and Werner's technical result [3] with probability exponentially close to one viz. with probability at least $1 - 2^{-m'}$, a Haar random subspace W of dimension m' satisfies $\|MM^\dagger - \frac{\mathbb{1}}{k}\|_2 < \frac{2}{k}$ for all $M \in W$. We will see below how this implies the existence of a channel with strictly subadditive minimum output von Neumann entropy.

In Step III, we define the function $f : \mathbb{S}_{\mathbb{C}^{k^3}} \to \mathbb{R}$ as $f(M) := \|MM^\dagger - \frac{\mathbb{1}}{k}\|_2^2$ i.e. this f is the square of the f defined in the last paragraph. Now, f is a balanced polynomial of degree $a = 2$ and $1 < \alpha(f) < k^6$ as can be seen by considering $f(J)$ where J is the $k \times k^2$ all ones matrix. The function f has global Lipschitz constant $L_1 = 4$ since

$$\begin{aligned} |f(M) - f(N)| &\leq |\|MM^\dagger - \frac{\mathbb{1}}{k}\|_2 - \|NN^\dagger - \frac{\mathbb{1}}{k}\|_2| \cdot |\|MM^\dagger - \frac{\mathbb{1}}{k}\|_2 + \|NN^\dagger - \frac{\mathbb{1}}{k}\|_2| \\ &\leq (\|M\|_\infty + \|N\|_\infty)(\|MM^\dagger - \frac{\mathbb{1}}{k}\|_2 + \|NN^\dagger - \frac{\mathbb{1}}{k}\|_2)\|M - N\|_2 \\ &\leq 4\|M - N\|_2. \end{aligned}$$

The mean μ of f under the Haar measure is less than $c_0^2 k^{-2}$ for the same universal constant c_0 [3, Corollary 7]. We use the notation of Theorem 21. Define $L_2 := 16k^{-3/2}$, $p(i) := i^3$ for all $i \in \mathbb{N}$. Then $C \leq 5$. Define the layers $\Omega_1, \Omega_2, \ldots,$ to be the subsets

$$\Omega_i := \left\{ M \in \mathbb{S}_{\mathbb{C}^{k^3}} : \|M\|_\infty \leq 2\sqrt{\frac{i}{k}}, \|MM^\dagger - \frac{\mathbb{1}}{k}\|_2 < \frac{2i}{k} \right\}.$$

It is easy to see that f restricted to Ω_i has local Lipschitz constant at most $L_2\sqrt{p(i)}$. Let $\lambda := k^{-2}$. Define $c_1 := 2^8\hat{c}_2$, $m = \hat{c}_2 k^2 < \hat{c}_1 k^2$, $c_2 := 2$, $c_3 := \ln 2$. By the previous two paragraphs, a Haar random subspace of dimension mi lies in Ω_i with probability at least $1 - c_2 e^{-c_3 mi}$. In particular, a Haar random subspace of dimension m lies in Ω_i with probability at least $1 - c_2 e^{-c_3 mi}$. Let

$$0 \leq \epsilon < \left(\frac{1}{16k^2}\right)^{2m} \frac{k^{9m} k^{-3m}}{k^{12m}} = (4k)^{-10\hat{c}_2 k^2}.$$

Theorem 21 tells us that there is a universal constant $\hat{c}_3$ such that for

$$m' := \hat{c}_3 k^2 \frac{\log\log k}{\log k},$$

with probability at least $1 - 3 \cdot 2^{-m'}$, a subspace W of dimension m' chosen from an ϵ-approximate $(4\hat{c}_2 k^2)$-design ν satisfies

$$f(M) = \|MM^\dagger - \frac{\mathbb{1}}{k}\|_2^2 < \frac{c_0^2}{k^2} + \frac{1}{k^2} = \frac{c_0^2+1}{k^2}$$

for all $M \in W$.

We shall now see how this result gives us a channel with strict subadditivity of minimum output von Neumann entropy. Consider the channel Φ corresponding to the subspace W. The output dimension is k. The input dimension is $\dim W = m'$. The Stinespring dilation of the channel Φ is the $k^3 \times k^3$ unitary matrix that defines the subspace W'. The subspace W' is obtained by taking the first m' columns of the unitary matrix. This unitary matrix is chosen uniformly at random from a $k^{-8\hat{c}_2 k^2}$-approximate unitary $(4\hat{c}_2 k^2)$-design. From Fact 3, we get

$$S_{\min}(\Phi) \geq \log k - k \max_{M \in W} f(M) \geq \log k - \frac{c_0^2+1}{k}.$$

And from Fact 4, with $d = k^2$, we get

$$\begin{aligned} S_{\min}(\Phi \otimes \bar{\Phi}) &\leq 2\log k - \frac{m'}{kd}\log k + O\left(\frac{m'}{kd}\log\frac{d}{m'} + \frac{1}{k}\right) \\ &= 2\log k - \frac{\hat{c}_3 \log\log k}{k} + O\left(\frac{(\log\log k)^2}{k\log k} + \frac{1}{k}\right) \\ &< S_{\min}(\Phi) + S_{\min}(\bar{\Phi}), \end{aligned}$$

for large enough k. Thus, we have shown that for large enough n approximate unitary $n^{2/3}$-designs give rise to channels exhibiting strict subadditivity of minimum output von Neumann entropy, implying that classical Holevo capacity of quantum channels can be superadditive.

If instead we were to use the existence of a Haar random subspace W of dimension $m := \hat{c}_2 k^2$ proved at the end of Step II above, we will recover Aubrun, Szarek and Werner's result on strict subadditivity of minimum output von Neumann entropy with probability exponentially close to one. Consider the channel Φ corresponding to the subspace W. The output dimension is k. The input dimension is $\dim W = m$. The Stinespring dilation of the channel Φ is the $k^3 \times k^3$ unitary matrix that defines the subspace W. The subspace W is obtained by taking the first m columns of a Haar random unitary matrix. From Fact 3, we get

$$S_{\min}(\Phi) \geq \log k - k \max_{M \in W} \|MM^\dagger - \frac{\mathbf{1}}{k}\|_2^2 \geq \log k - \frac{4}{k}.$$

And from Fact 4, with $d = k^2$, we get

$$\begin{aligned} S_{\min}(\Phi \otimes \bar{\Phi}) &\leq 2\log k - \frac{m}{kd}\log k + O\left(\frac{m}{kd}\log\frac{d}{m} + \frac{1}{k}\right) \\ &= 2\log k - \frac{\hat{c}_2 \log k}{k} + O\left(\frac{1}{k}\right) \\ &< S_{\min}(\Phi) + S_{\min}(\bar{\Phi}), \end{aligned}$$

for large enough k. Thus, we have shown that for large enough n Haar random unitaries give rise to channels exhibiting strict subadditivity of minimum output von Neumann entropy, implying that classical Holevo capacity of quantum channels can be superadditive. Observe that the counter example we get for additivity conjecture for classical Holevo capacity of quantum channels, when the channel is chosen from an approximate unitary t-design has weaker parameters than a channel chosen from Haar random unitaries. Nevertheless, as explained in the introduction our work is the first partial derandomisation of a construction of quantum channels violating additivity of classical Holevo capacity.

References

1 G. G. Amosov, A. S. Holevo, and R. F. Werner. On some additivity problems in quantum information theory. *arXiv e-prints*, pages math–ph/0003002, March 2000. arXiv:math-ph/0003002.

2 Anderson, G., Guionnet, A., and Zeitouni, O. *An introduction to random matrices*. Cambridge University Press, 2009.

3 Guillaume Aubrun, Stanisław Szarek, and Elisabeth Werner. Hastings's Additivity Counterexample via Dvoretzky's Theorem. *Communications in Mathematical Physics*, 305(1):85–97, 2010. doi:10.1007/s00220-010-1172-y.

4 Guillaume Aubrun, Stanisław Szarek, and Elisabeth Werner. Nonadditivity of Rényi entropy and Dvoretzky's theorem. *Journal of Mathematical Physics*, 51(2):022102, 2010. doi:10.1063/1.3271044.

5 Charles H. Bennett, David P. DiVincenzo, John A. Smolin, and William K. Wootters. Mixed-state entanglement and quantum error correction. *Phys. Rev. A*, 54:3824–3851, November 1996. doi:10.1103/PhysRevA.54.3824.

6 Brandão, F., Harrow, A., and Horodecki, M. Local Random Quantum Circuits are Approximate Polynomial-Designs. *Communications in Mathematical Physics*, 346(2):397–434, 2016.

7 Aryeh Dvoretzky. Some results on convex bodies and Banach spaces. In *Proc. Internat. Sympos. Linear Spaces (Jerusalem, 1960)*, pages 123–160. Jerusalem Academic Press, Jerusalem; Pergamon, Oxford, 1961.

8 Akio Fujiwara and Takashi Hashizumé. Additivity of the capacity of depolarizing channels. *Physics Letters A*, 299(5):469–475, 2002. doi:10.1016/S0375-9601(02)00735-1.

9 Yehoram Gordon. Some inequalities for Gaussian processes and applications. *Israel Journal of Mathematics*, 50(4):265–289, December 1985. doi:10.1007/BF02759761.

10 M. B. Hastings. Superadditivity of communication capacity using entangled inputs. *Nature Physics*, 5(4):255–257, 2009. doi:10.1038/nphys1224.

11 Patrick Hayden and Andreas Winter. Counterexamples to the Maximal p-Norm Multiplicativity Conjecture for all $p > 1$. *Communications in Mathematical Physics*, 284(1):263–280, October 2008. doi:10.1007/s00220-008-0624-0.

12 C. King. The capacity of the quantum depolarizing channel. *IEEE Transactions on Information Theory*, 49(1):221–229, January 2003. doi:10.1109/TIT.2002.806153.

13 Christopher King. Additivity for unital qubit channels. *Journal of Mathematical Physics*, 43(10):4641–4653, 2002. doi:10.1063/1.1500791.

14 Christopher King, Keiji Matsumoto, Michael Nathanson, and Mary Beth Ruskai. Properties of Conjugate Channels with Applications to Additivity and Multiplicativity. *arXiv e-prints*, pages quant–ph/0509126, September 2005. arXiv:quant-ph/0509126.

15 G. Kuperberg. Numerical Cubature from Archimedes' Hat-Box Theorem. *SIAM Journal on Numerical Analysis*, 44(3):908–935, 2006.

16 R. A. Low. Large deviation bounds for k-designs. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 465(2111):3289–3308, May 2009. doi:10.1098/rspa.2009.0232.

17 V. Milman. A new proof of the theorem of A. Dvoretzky on sections of convex bodies. *Func. Anal. Appl.*, 5:28–37, 1992. English translation.

18 L. Mirsky. Symmetric gauge functions and unitarily invariant norms. *Quarterly Journal of Mathematics Oxford*, 11(1):50–59, 1960.

19 Aditya Nema and Pranab Sen. Approximate unitary designs give rise to quantum channels with super additive classical capacity. *arXiv e-prints*, page arXiv:1902.10808, February 2019. arXiv:1902.10808.

20 Susumu Osawa and Hiroshi Nagaoka. Numerical Experiments on The Capacity of Quantum Channel with Entangled Input States. *IEICE Transactions on Fundamentals of Electronics Communications and Computer Sciences*, E84A, August 2000.

21 Gilles Pisier. *The volume of convex bodies and Banach space geometry*. Cambridge University Press, 1989.

22 A. A. Pomeransky. Strong superadditivity of the entanglement of formation follows from its additivity. *Phys. Rev. A*, 68:032317, September 2003. doi:10.1103/PhysRevA.68.032317.

23 G. Schechtman. A remark concerning the dependence on ϵ in Dvoretzky's theorem. In *Geometric Aspects of Functional Analysis Israel Seminar (GAFA) 1987–88*, pages 274–277. Springer, Berlin, Heidelberg, 1988. doi:10.1007/BFb0090046.

24 P. Sen. Efficient quantum tensor product expanders and unitary t-designs via the zigzag product. arXiv preprint, 2018. arXiv:1808.10521.

25 Peter W. Shor. Additivity of the classical capacity of entanglement-breaking quantum channels. *Journal of Mathematical Physics*, 43(9):4334–4340, 2002. doi:10.1063/1.1498000.

26 Peter W. Shor. Equivalence of Additivity Questions in Quantum Information Theory. *Communications in Mathematical Physics*, 246(3):473–473, April 2004. doi:10.1007/s00220-004-1071-1.

27 S. Vajda, Claude E. Shannon, and Warren Weaver. The Mathematical Theory of Communication. *The Mathematical Gazette*, 34(310):312, 1950. doi:10.2307/3611062.

28 R. Vershynin. *High dimensional probability*. Cambridge University Press, 2018.

A Proofs of Lemmas used in stratified analysis

Proof of Lemma 16. By Fact 8, there is a circled function Y' that agrees with Y on Ω and is L_2-Lipschitz on all of $\mathbb{S}_{\mathbb{C}^n}$. Define correlated random variables Y'_x, Y'_y in the natural manner. Then using Fact 11, we get

$$\begin{aligned}
&\Pr_{U\sim\text{Haar}}[|Y_x - Y_y| > \lambda] \\
&= \Pr_{U\sim\text{Haar}}[(Ux,Uy) \in \Omega\times\Omega] \cdot \Pr_{U\sim\text{Haar}}[|Y_x - Y_y| > \lambda | (Ux,Uy) \in \Omega\times\Omega] \\
&\quad + \Pr_{U\sim\text{Haar}}[(Ux,Uy) \notin \Omega\times\Omega] \cdot \Pr_{U\sim\text{Haar}}[|Y_x - Y_y| > \lambda | (Ux,Uy) \notin \Omega\times\Omega] \\
&= \Pr_{U\sim\text{Haar}}[(Ux,Uy) \in \Omega\times\Omega] \cdot \Pr_{U\sim\text{Haar}}[|Y'_x - Y'_y| > \lambda | (Ux,Uy) \in \Omega\times\Omega] \\
&\quad + \Pr_{U\sim\text{Haar}}[(Ux,Uy) \notin \Omega\times\Omega] \cdot \Pr_{U\sim\text{Haar}}[|Y_x - Y_y| > \lambda | (Ux,Uy) \notin \Omega\times\Omega] \\
&\leq \Pr_{U\sim\text{Haar}}[|Y'_x - Y'_y| > \lambda] + 2\Pr_{z\sim\text{Haar}}[z \in \Omega^c] \\
&\leq 2\exp(-\frac{n\lambda^2}{8L_2^2\|x-y\|_2^2}) + 2\Pr_{z\sim\text{Haar}}[z \in \Omega^c].
\end{aligned}$$

This finishes the proof of the lemma. ◀

Proof of Lemma 17. Since $Z_x - Z_y$ is a balanced polynomial in the entries of the unitary matrix U, from Fact 15 we have

$$\mathbb{E}_{U\sim\nu}[|Z_x - Z_y|^{2r}] \overset{\text{a}}{\leq} \mathbb{E}_{U\sim\text{Haar}}[|Z_x - Z_y|^{2r}] + \frac{\epsilon\alpha(Z)^{2r}}{n^t}.$$

By choosing ϵ small enough to satisfy the constraint above, we get $\frac{\epsilon\alpha(Z)^{2r}}{n^t} \overset{\text{b}}{\leq} \left(\frac{4rL_2^2\|x-y\|_2^2}{n}\right)^r$. Combining (a) and (b) gives

$$\mathbb{E}_{U\sim\nu}[|Z_x - Z_y|^{2r}] \overset{\text{c}}{\leq} \mathbb{E}_{U\sim\text{Haar}}(|Z_x - Z_y|^{2r}) + \left(\frac{4rL_2^2\|x-y\|_2^2}{n}\right)^r.$$

Now we find $\mathbb{E}_{U\sim\text{Haar}}[|Z_x - Z_y|^{2r}]$.

By Fact 8, there is a circled function Y' that agrees with Y on Ω and is L_2-Lipschitz on all of $\mathbb{S}_{\mathbb{C}^n}$. Define correlated random variables Y'_x, Y'_y in the natural manner. Then using Fact 11, we get

$$\begin{aligned}
&\Pr_{U\sim\text{Haar}}[|Y_x - Y_y| > \lambda] \\
&= \Pr_{U\sim\text{Haar}}[(Ux,Uy) \in \Omega\times\Omega] \cdot \Pr_{U\sim\text{Haar}}[|Y_x - Y_y| > \lambda | (Ux,Uy) \in \Omega\times\Omega] \\
&\quad + \Pr_{U\sim\text{Haar}}[(Ux,Uy) \notin \Omega\times\Omega] \cdot \Pr_{U\sim\text{Haar}}[|Y_x - Y_y| > \lambda | (Ux,Uy) \notin \Omega\times\Omega] \\
&= \Pr_{U\sim\text{Haar}}[(Ux,Uy) \in \Omega\times\Omega] \cdot \Pr_{U\sim\text{Haar}}[|Y'_x - Y'_y| > \lambda | (Ux,Uy) \in \Omega\times\Omega] \\
&\quad + \Pr_{U\sim\text{Haar}}[(Ux,Uy) \notin \Omega\times\Omega] \cdot \Pr_{U\sim\text{Haar}}[|Y_x - Y_y| > \lambda | (Ux,Uy) \notin \Omega\times\Omega] \\
&\leq \Pr_{U\sim\text{Haar}}[|Y'_x - Y'_y| > \lambda] + 2\Pr_{z\sim\text{Haar}}[z \in \Omega^c] \\
&\leq 2\exp(-\frac{n\lambda^2}{8L_2^2\|x-y\|_2^2}) + 2\Pr_{z\sim\text{Haar}}[z \in \Omega^c].
\end{aligned}$$

This finishes the proof of the lemma. ◀

Proof of Lemma 18. Since $Y_x - Y_y$ is a balanced polynomial in the entries of the unitary matrix U, from Fact 15 we have

$$\mathbb{E}_{U\sim\nu}[|Y_x - Y_y|^{2r}] \overset{\text{a}}{\leq} \mathbb{E}_{U\sim\text{Haar}}[|Y_x - Y_y|^{2r}] + \frac{\epsilon\alpha(Y)^{2r}}{n^t}.$$

By choosing ϵ small enough to satisfy the constraint above, we get $\frac{\epsilon\alpha(Y)^{2r}}{n^t} \overset{\text{b}}{\leq} \left(\frac{4rL_2^2\|x-y\|_2^2}{n}\right)^r$. Combining (a) and (b) gives

$$\mathbb{E}_{U\sim\nu}[|Y_x - Y_y|^{2r}] \overset{\text{c}}{\leq} \mathbb{E}_{U\sim\text{Haar}}(|Y_x - Y_y|^{2r}) + \left(\frac{4rL_2^2\|x-y\|_2^2}{n}\right)^r.$$

Now we find $\mathbb{E}_{U\sim\text{Haar}}[|Y_x - Y_y|^{2r}]$. Since Y is a balanced polynomial, it is circled. By Fact 8, there is a circled function Y' such that Y' agrees with Y on Ω and Y' is L_2-Lipschitz on all of $\mathbb{S}_{\mathbb{C}^n}$. Define correlated random variables Y'_x, Y'_y in the natural manner. Then

$$\begin{aligned}
&\mathbb{E}_{U\sim\text{Haar}}[|Y_x - Y_y|^{2r}] \\
&= \Pr_{U\sim\text{Haar}}[(Ux,Uy) \in \Omega\times\Omega] \cdot \mathbb{E}_{U\sim\text{Haar}}[|Y_x - Y_y|^{2r} | (Ux,Uy) \in \Omega\times\Omega] \\
&\quad + \Pr_{U\sim\text{Haar}}[(Ux,Uy) \notin \Omega\times\Omega] \cdot \mathbb{E}_{U\sim\text{Haar}}[|Y_x - Y_y|^{2r} | (Ux,Uy) \notin \Omega\times\Omega] \\
&= \Pr_{U\sim\text{Haar}}[(Ux,Uy) \in \Omega\times\Omega] \cdot \mathbb{E}_{U\sim\text{Haar}}[|Y'_x - Y'_y|^{2r} | (Ux,Uy) \in \Omega\times\Omega] \\
&\quad + \Pr_{U\sim\text{Haar}}[(Ux,Uy) \notin \Omega\times\Omega] \cdot \mathbb{E}_{U\sim\text{Haar}}[|Y_x - Y_y|^{2r} | (Ux,Uy) \notin \Omega\times\Omega] \\
&\overset{\text{d}}{\leq} \mathbb{E}_{U\sim Haar}[|Y'_x - Y'_y|^{2r}] + 2\Pr_{z\sim\text{Haar}}[z \in \Omega^c] \cdot (L_1^2\|x-y\|_2^2)^r.
\end{aligned}$$

Now we find $\mathbb{E}_{U\sim\text{Haar}}[|Y'_x - Y'_y|^{2r}]$ using Fact 11 and Low's method [16, Lemma 3.3].

$$\begin{aligned}
&\mathbb{E}_{U\sim\text{Haar}}[|Y'_x - Y'_y|^{2r}] \\
&= \int_0^\infty \Pr_{U\sim\text{Haar}}[|Y'_x - Y'_y|^{2r} > \lambda]\, d\lambda \;=\; \int_0^\infty \Pr_{U\sim\text{Haar}}[|Y'_x - Y'_y| > \lambda^{1/(2r)}]\, d\lambda \\
&\leq 2\int_0^\infty \exp(-\frac{n\lambda^{1/r}}{8L_2^2\|x-y\|_2^2})\, d\lambda \overset{e}{\leq} 2\left(\frac{4rL_2^2\|x-y\|_2^2}{n}\right)^r.
\end{aligned}$$

Combining inequalities (d) and (e), we have

$$\mathbb{E}_{U\sim\text{Haar}}[|Y_x - Y_y|^{2r}] \leq 2\left(\frac{4rL_2^2\|x-y\|_2^2}{n}\right)^r + 2\Pr_{z\sim\text{Haar}}[z\in\Omega^c]\cdot(L_1^2\|x-y\|_2^2)^r.$$

Further combining with (c) gives us the desired conclusion of the lemma. ◀

B Strict subadditivity of minimum output Rényi p-entropy for approximate t-designs

We now give a general proposition showing how to approximate a continuous non-decreasing function by a polynomial of moderate degree. The proof can be found in the full version in [19].

▶ **Proposition 22.** *Let $f : [0, A] \to [0, 1]$ be a continuous non-decreasing onto function that has left and right derivatives everywhere. Define the* global Lipschitz constant $L := \max_{y\in[0,A]} f'(y)$, *where henceforth we use $f'(y)$ to denote the maximum of the the left and right derivatives of f at y. Fix $0 < \epsilon < 1$. Define the* ϵ-smoothed local Lipschitz constant at x,

$$L_x^\epsilon := \max_{y\in f^{-1}((f(x)-\epsilon, f(x)+\epsilon))} f'(y).$$

Let n be the minimum positive odd integer satisfying $mA \leq \frac{\epsilon^{\frac{1}{n}}\sqrt{n}}{2}$, where $m := \frac{2L}{\epsilon}\sqrt{\ln \epsilon^{-2}}$. Define $m_x := \frac{2L_x^\epsilon}{\epsilon}\sqrt{\ln\epsilon^{-2}}$. Then there is a polynomial $p(x)$ of degree at most $2n+1$ such that

$$p(x) - 2\epsilon \leq f(x) \leq p(x) + 3\epsilon, \quad -m\epsilon^2 < p'(x) < \epsilon m_x + m\epsilon^2, \quad \forall x \in [0, A].$$

Moreover the sum of absolute values of the coefficients of $p(x)$, denoted by $\alpha(p(x))$, is at most $e^{2((A+1)m)^2}$.

In this section, we apply Proposition 22 and Theorem 21 in order to show that channels with approximate $p^3 n^{\frac{4}{9p}+\frac{5}{9}+\frac{2p}{3}}\log n$-design unitary Stinespring dilations exhibit strict subadditivity of minimum output Rényi p-entropy for $p > 1$ with exponentially high probability close to one.

Let k be a positive integer. Consider the sphere $\mathbb{S}_{\mathbb{C}^{k^3}}$. Define the $k \times k^2$ matrix M to be the rearrangment of a k^3-tuple from $\mathbb{S}_{\mathbb{C}^{k^3}}$. Note that the ℓ_2-norm on $\mathbb{C}^{k^3}$ is the same as the Frobenius norm on $\mathbb{C}^{k\times k^2}$. Let $p > 1$.

In Step I, we define the function $f : \mathbb{S}_{\mathbb{C}^{k^3}} \to \mathbb{R}$ as $f(M) := \|M\|_{2p}$. The function f has global Lipschitz constant $L_1 = 1$ since

$$|f(M) - f(N)| \leq \|M - N\|_{2p} \leq \|M - N\|_2.$$

For large enough k the mean μ of f, under the Haar measure, is less than $2k^{\frac{1}{2p}-\frac{1}{2}}$ [4, Section VIII], [3, Corollary 7]. We use the notation of Theorem 20. Define $L_2 := 1$, $p(i) := 1$ for all $i \in \mathbb{N}$. Then $C < 2$. Define the layers $\Omega_1, \Omega_2, \ldots,$ to be all of $\mathbb{S}_{\mathbb{C}^{k^3}}$. Let j, $4 \leq j \leq k$ be

a positive integer. Let $\lambda_j := j^{\frac{1}{2}} k^{\frac{1}{2p}-\frac{1}{2}}$. Define $c_1 := 1$, $m = k^{2+\frac{1}{p}}$, $c_2 := 0$, $c_3 := 1$. Trivially, a Haar random subspace of dimension mj lies in Ω_i with probability at least $1 - c_2 e^{-c_3 mji}$. Theorem 20 tells us that there is a universal constant $\hat{c}_1$ such that for $m' := \hat{c}_1 k^{2+\frac{1}{p}}$, with probability at least $1 - 2^{-m'j}$, a Haar random subspace W of dimension $m'j$ satisfies

$$\|M\|_\infty \leq \|M\|_{2p} < 2k^{\frac{1}{2p}-\frac{1}{2}} + j^{\frac{1}{2}} k^{\frac{1}{2p}-\frac{1}{2}} < 2j^{\frac{1}{2}} k^{\frac{1}{2p}-\frac{1}{2}}$$

for all $M \in W$. In particular, with probability at least $1 - 2^{-\hat{c}_1 jk^{\frac{4}{3p}+\frac{5}{3}} (\log k)^{-1}}$, a Haar random subspace W of dimension $\hat{c}_1 jk^{\frac{4}{3p}+\frac{5}{3}} (\log k)^{-1}$ satisfies

$$\|M\|_\infty \leq \|M\|_{2p} < 2j^{\frac{1}{2}} k^{\frac{1}{2p}-\frac{1}{2}}$$

for all $M \in W$.

Let j, $4 \leq j \leq k$ be a positive integer. Define the function $f : [0,1] \to [0,1]$ as $f(x) := x^p$. Set $\epsilon := k^{-p}$ in Proposition 22. Let n be the minimum positive odd integer satisfying $2pk^p \sqrt{\ln k^{2p}} \leq \frac{k^{-\frac{p}{n}} \sqrt{n}}{2}$; $n < 2^7 p^3 k^{2p} \log k$. Proposition 22 implies that there is a polynomial $p(x)$ of degree at most $2n + 1 < 2^9 p^3 k^{2p} \log k$ such that

$$\begin{array}{ll} p(x) - 2k^{-p} \leq x^p \leq p(x) + 3k^{-p}, & \forall x \in [0,1], \\ |p'(x)| < 4p(j+1)^{p-1} \sqrt{\ln k^{2p}} k^{\frac{5}{3}-\frac{2}{3p}-p}, & \forall x \in [0, jk^{\frac{2}{3p}-1}], \\ |p'(x)| < 4p(5j)^{p-1} \sqrt{\ln k^{2p}} k^{2-p-\frac{1}{p}}, & \forall x \in (jk^{\frac{2}{3p}-1}, 5jk^{\frac{1}{p}-1}], \\ |p'(x)| < 4p\sqrt{\ln k^{2p}}, & \forall x \in (5jk^{\frac{1}{p}-1}, 1]. \end{array} \tag{3}$$

Also, Proposition 22 guarantees that $\alpha(p(x)) < e^{2^7 p^3 k^{2p} \log k}$.

In Step II, we define the function $f : \mathbb{S}_{\mathbb{C}^{k^3}} \to \mathbb{R}$ as $f(M) := \mathrm{Tr}\,[p(MM^\dagger)]$, where p is the polynomial defined in Equation 3. Now, f is a balanced polynomial of degree $a = 2n + 1 < 2^9 p^3 k^{2p} \log k$ and

$$\alpha(f) = \mathrm{Tr}\,[p(JJ^\dagger)] = k^3 \alpha(p(x)) < e^{2^8 p^3 k^{2p} \log k},$$

where J is the $k \times k^2$ all ones matrix. For a $k \times k$ matrix X, define $\mathrm{Sing}(X)$ to be the $k \times k$ diagonal matrix consisting of the singular values of X arranged in decreasing order. The function f has global Lipschitz constant $L_1 = 2^4 p^{3/2} \sqrt{\log k}$ since

$$\begin{aligned} |f(M) - f(N)| &= |\mathrm{Tr}\,[p(\mathrm{Sing}(M)^2)] - \mathrm{Tr}\,[p(\mathrm{Sing}(N)^2)]| = |\mathrm{Tr}\,[p(\mathrm{Sing}(M)^2) - p(\mathrm{Sing}(N)^2)]| \\ &\leq 8p^{3/2} \sqrt{\log k} \cdot \|\mathrm{Sing}(M)^2 - \mathrm{Sing}(N)^2\|_1 \\ &\leq 8p^{3/2} \sqrt{\log k} \cdot \|\mathrm{Sing}(M) - \mathrm{Sing}(N)\|_2 \cdot \|\mathrm{Sing}(M) + \mathrm{Sing}(N)\|_2 \\ &\leq 2^{7/2} p^{3/2} \sqrt{\log k} \cdot \|\mathrm{Sing}(M) - \mathrm{Sing}(N)\|_2 \cdot \sqrt{\|M\|_2^2 + \|N\|_2^2} \\ &\leq 2^4 p^{3/2} \sqrt{\log k} \cdot \|M - N\|_2. \end{aligned}$$

Above, the first inequality follows from Equation 3, the second inequality is Cauchy-Schwarz and the last inequality follows from [18, Section 4]. By setting $j = 4$ in Step I, we conclude that the mean μ of f under the Haar measure is less than $2^{4p} k^{1-p}$. We use the notation of Theorem 21. Let $\lambda := k^{1-p}$. Define

$$L_2 := 2^{4p+3} p^{3/2} \sqrt{\log k} \cdot k^{\frac{5}{3}-p-\frac{2}{3p}},$$

$p(i) := (i+4)^{2p-1}$ for all $i \in \mathbb{N}$. Then $C \leq p^{2p}$. Define the layers $\Omega_1, \Omega_2, \ldots,$ to be the subsets

$$\Omega_i := \left\{ M \in \mathbb{S}_{\mathbb{C}^{k^3}} : \|M\|_{2p} \leq 2(i+3)^{\frac{1}{2}} k^{\frac{1}{2p}-\frac{1}{2}} \right\}.$$

We will now show that f restricted to Ω_i has local Lipschitz constant at most $L_2\sqrt{p(i)}$. Note that for any $M \in \Omega_i$, $\|M\|_\infty \leq 2(i+3)^{\frac{1}{2}}k^{\frac{1}{2p}-\frac{1}{2}}$. Let B denote the number of singular values of M larger than $(i+3)^{\frac{1}{2}}k^{\frac{1}{3p}-\frac{1}{2}}$. Let $b_1, \ldots b_k$ be the singular values of M in descending order. Then

$$2^{2p}(i+3)^p k^{1-p} \geq \|M\|_{2p}^{2p} \geq \sum_{i=1}^{B} b_i^{2p} \geq \left(\sum_{i=1}^{B} b_i^2\right)(i+3)^{p-1}k^{\frac{5}{3}-\frac{2}{3p}-p},$$

which gives $\sum_{i=1}^{B} b_i^2 \leq 2^{2p}(i+3)k^{\frac{2}{3p}-\frac{2}{3}}$. Let C denote the number of singular values of N larger than $(i+3)^{\frac{1}{2}}k^{\frac{1}{3p}-\frac{1}{2}}$. Without loss of generality, $B \geq C$. Restricting M, N to belong to Ω_i, we get from Equation 3 that

$$\begin{aligned}
&|f(M)-f(N)| \\
&= |\mathrm{Tr}\left[p(\mathrm{Sing}(M)^2) - p(\mathrm{Sing}(N)^2)\right]| \\
&\leq \sum_{i=1}^{C}|p(b_i^2)-p(c_i^2)| + \sum_{i=C+1}^{B}|p(b_i^2)-p(c_i^2)| + \sum_{i=B+1}^{k}|p(b_i^2)-p(c_i^2)| \\
&\leq 8p^{3/2}(5(i+3))^{p-1}\sqrt{\log k}\cdot k^{2-p-\frac{1}{p}}\sum_{i=1}^{C}|b_i^2-c_i^2| \\
&\quad + 8p^{3/2}(5(i+3))^{p-1}\sqrt{\log k}\cdot k^{2-p-\frac{1}{p}}\sum_{i=C+1}^{B}|b_i^2-c_i^2| \\
&\quad + 8p^{3/2}((i+4))^{p-1}\sqrt{\log k}\cdot k^{\frac{5}{3}-p-\frac{2}{3p}}\sum_{i=B+1}^{k}|p(b_i^2)-p(c_i^2)| \\
&\leq 8p^{3/2}(5(i+3))^{p-1}\sqrt{\log k}\cdot k^{2-p-\frac{1}{p}}\sqrt{\sum_{i=1}^{C}(b_i-c_i)^2}\cdot\sqrt{\sum_{i=1}^{C}(b_i+c_i)^2} \\
&\quad + 8p^{3/2}(5(i+3))^{p-1}\sqrt{\log k}\cdot k^{2-p-\frac{1}{p}}\sqrt{\sum_{i=C+1}^{B}(b_i-c_i)^2}\cdot\sqrt{\sum_{i=C+1}^{B}(b_i+c_i)^2} \\
&\quad + 8p^{3/2}((i+4))^{p-1}\sqrt{\log k}\cdot k^{\frac{5}{3}-p-\frac{2}{3p}}\sqrt{\sum_{i=B+1}^{k}(b_i-c_i)^2}\cdot\sqrt{\sum_{i=B+1}^{k}(b_i+c_i)^2} \\
&\leq 2^{7/2}p^{3/2}(5(i+3))^{p-1}\sqrt{\log k}\cdot k^{2-p-\frac{1}{p}}\sqrt{\sum_{i=1}^{k}(b_i-c_i)^2}\cdot\sqrt{\sum_{i=1}^{C}(b_i^2+c_i^2)} \\
&\quad + 2^{7/2}p^{3/2}(5(i+3))^{p-1}\sqrt{\log k}\cdot k^{2-p-\frac{1}{p}}\sqrt{\sum_{i=1}^{k}(b_i-c_i)^2}\cdot\sqrt{\sum_{i=C+1}^{B}(b_i^2+c_i^2)} \\
&\quad + 2^{7/2}p^{3/2}((i+4))^{p-1}\sqrt{\log k}\cdot k^{\frac{5}{3}-p-\frac{2}{3p}}\sqrt{\sum_{i=1}^{k}(b_i-c_i)^2}\cdot\sqrt{\sum_{i=1}^{k}(b_i^2+c_i^2)} \\
&\leq 2^4p^{3/2}2^p5^{p-1}(i+3)^{p-\frac{1}{2}}\sqrt{\log k}\cdot k^{2-p-\frac{1}{p}}\cdot k^{\frac{1}{3p}-\frac{1}{3}}\cdot\|\mathrm{Sing}(M)-\mathrm{Sing}(N)\|_2 \\
&\quad + 2^4p^{3/2}2^p5^{p-1}(i+3)^{p-\frac{1}{2}}\sqrt{\log k}\cdot k^{2-p-\frac{1}{p}}\cdot k^{\frac{1}{3p}-\frac{1}{3}}\cdot\|\mathrm{Sing}(M)-\mathrm{Sing}(N)\|_2 \\
&\quad + 2^4p^{3/2}((i+4))^{p-1}\sqrt{\log k}\cdot k^{\frac{5}{3}-p-\frac{2}{3p}}\|\mathrm{Sing}(M)-\mathrm{Sing}(N)\|_2 \\
&\leq 2^6p^{3/2}2^p5^{p-1}(i+4)^{p-\frac{1}{2}}\sqrt{\log k}\cdot k^{\frac{5}{3}-p-\frac{2}{3p}}\cdot\|\mathrm{Sing}(M)-\mathrm{Sing}(N)\|_2 \\
&\leq 2^{4p+3}p^{3/2}(i+4)^{p-\frac{1}{2}}\sqrt{\log k}\cdot k^{\frac{5}{3}-p-\frac{2}{3p}}\cdot\|M-N\|_2.
\end{aligned}$$

This completes the proof of the claim above that f restricted to Ω_i has local Lipschitz constant at most $L_2\sqrt{p(i)}$. Define $c_1 := 2^{8p+6}p^3\hat{c}_1$, $m = \hat{c}_1 k^{\frac{4}{3p}+\frac{5}{3}}(\log k)^{-1}$, $c_2 := 1$, $c_3 := \ln 2$. By Step I, a Haar random subspace of dimension mi lies in Ω_i with probability at least $1 - c_2 e^{-c_3 mi}$. Let

$$0 \leq \epsilon < \left(\frac{k^{1-p}}{4L_1}\right)^{2m} \frac{k^{3(2a-1)m}5^{(2p-1)m}L_2^{2m}}{\alpha(f)^{2m}}.$$

Theorem 21 tells us that there is a universal constant $\hat{c}_3$ such that for

$$m' := \hat{c}_3 k^{\frac{4}{3p}+\frac{5}{3}} \frac{\log\log k}{(\log k)^2},$$

with probability at least $1 - 2 \cdot 2^{-m'}$, a subspace W of dimension m' chosen from an ϵ-approximate $(2am)$-design ν satisfies

$$f(M) = \operatorname{Tr}\left[p(MM^\dagger)\right] < 2^{4p}k^{1-p} + k^{1-p} < 2^{4p+1}k^{1-p}$$

for all $M \in W$. By Equation 3, this implies that

$$\operatorname{Tr}\left[(MM^\dagger)^p\right] < \operatorname{Tr}\left[p(MM^\dagger)\right] + 3k^{1-p} < 2^{4p+3}k^{1-p}$$

for all $M \in W$. In other words, $\|M\|_{2p}^2 < 2^7 k^{\frac{1}{p}-1}$ for all $M \in W$.

We shall now see how this result gives us a channel with strict supermultiplicativity of the $\|\cdot\|_{1\to p}$-norm or equivalently, strict subadditivity of minimum output Rényi p-entropy for $p > 1$. Consider the channel Φ corresponding to the subspace W. The output dimension is k. The input dimension is $\dim W = m'$. The Stinespring dilation of the channel Φ is the $k^3 \times k^3$ unitary matrix that defines the subspace W'. The subspace W' is obtained by taking the first m' columns of the unitary matrix. This unitary matrix is chosen uniformly at random from an ϵ-approximate unitary $\hat{c}_1 2^{10}p^3 k^{\frac{4}{3p}+\frac{5}{3}+2p}\log k$-design. From Equation 1, we get

$$\|\Phi\|_{1\to p} = \max_{M\in W:\|M\|_2=1} \|M\|_{2p}^2 \leq 2^7 k^{\frac{1}{p}-1}.$$

From Fact 4,

$$\|\Phi \otimes \bar{\Phi}\|_{1\to p} \geq \frac{m'}{k^3} = \hat{c}_3 k^{\frac{4}{3p}-\frac{4}{3}} \frac{\log\log k}{(\log k)^2} > (\|\Phi\|_{1\to p})^2$$

for large enough k. Thus, we have shown that for large enough n approximate unitary $p^3 n^{\frac{4}{9p}+\frac{5}{9}+\frac{2p}{3}}\log n$-designs give rise to channels exhibiting strict subadditivity of minimum output Rényi p-entropy for any $p > 1$.

Parameterization of Tensor Network Contraction

Bryan O'Gorman
Berkeley Quantum Information & Computation Center, University of California, Berkeley, CA, USA
Quantum Artificial Intelligence Laboratory, NASA Ames, Moffett Field, CA, USA
bogorman@berkeley.edu

Abstract

We present a conceptually clear and algorithmically useful framework for parameterizing the costs of tensor network contraction. Our framework is completely general, applying to tensor networks with arbitrary bond dimensions, open legs, and hyperedges. The fundamental objects of our framework are rooted and unrooted contraction trees, which represent classes of contraction orders. Properties of a contraction tree correspond directly and precisely to the time and space costs of tensor network contraction. The properties of rooted contraction trees give the costs of parallelized contraction algorithms. We show how contraction trees relate to existing tree-like objects in the graph theory literature, bringing to bear a wide range of graph algorithms and tools to tensor network contraction. Independent of tensor networks, we show that the edge congestion of a graph is almost equal to the branchwidth of its line graph.

2012 ACM Subject Classification Theory of computation → Fixed parameter tractability; Theory of computation → Quantum information theory; Theory of computation → Graph algorithms analysis

Keywords and phrases tensor networks, parameterized complexity, tree embedding, congestion

Digital Object Identifier 10.4230/LIPIcs.TQC.2019.10

Funding The author was supported by a NASA Space Technology Research Fellowship.

Acknowledgements The author thanks Benjamin Villalonga for motivating this work, useful discussions, and feedback on the manuscript.

1 Introduction

Tensor networks are widely used in chemistry and physics. Their graphical structure provides an effective way for expressing and reasoning about quantum states and circuits. As a model for quantum states, they have been very successful in expressing ansatzes in variational algorithms (e.g., PEPS, MPS, and MERA). As a model for quantum circuits, they have been used in state-of-the-art simulations [27, 19, 18, 20, 24]. In the other direction, quantum circuits can also simulate tensor networks, in the sense that (additively approximate) tensor network contraction is complete for quantum computation [3].

The essential computation in the application of tensor networks is tensor network contraction, i.e., computing the single tensor represented by a tensor network. Tensor network contraction is #P-hard in general [6] but fixed-parameter tractable. Markov and Shi [22] defined the contraction complexity of a tensor network and showed that contraction can be done in time that scales exponentially only in the treewidth of the line graph of the tensor network. Given a tree decomposition of the line graph of a tensor network, a contraction order can be found such that the contraction takes time exponential in the width of the decomposition, and vice versa. However, the translation between contraction orders and tree decompositions does not account for polynomial prefactors. This is acceptable in theory, where running times of $O(n2^n)$ and of $O(2^n)$ are both "exponential"; in practice, the difference between $\Theta(n2^n)$ and $\Theta(2^n)$ can be the difference between feasible and infeasible.

We give an alternative characterization of known results in terms of tree embeddings of the tensor network rather than tree decompositions of the line graph thereof. In this context, we call such tree embeddings *contraction trees*. While one can efficiently interconvert between a contraction tree of a tensor network and a tree decomposition of the line graph, contraction trees exactly model the matrix multiplications done by a contraction algorithm in an abstract way. That is, the time complexity of contraction is exactly and directly expressed as a property of contraction trees, in contrast to tree decompositions of line graphs, which only capture the exponent. Our approach is thus more intuitive and precise, and easily applies to tensor networks with arbitrary bond dimensions and open legs.

We show that contraction trees also capture the space needed by a matrix-multiplication-based contraction algorithm. In practice, space often competes with time as the limiting constraint. Even further, we can express the time used by parallel algorithms as a property of *rooted contraction trees*, which are to contraction orders as partial orders are to total orders.

In a contraction tree, tensors are assigned to the leaves and each wire is "routed" through the tree from one leaf to another. The congestion of a vertex of the contraction tree is the number of such routings that pass through it, and similarly for the congestion of an edge. The vertex congestion of a graph G, denoted $\mathrm{vc}(G)$, is the minimum over contraction trees of the maximum congestion of a vertex, and similarly for the edge congestion, denoted $\mathrm{ec}(G)$. Formally, our main results are the following two theorems.

▶ **Theorem 1.** *A tensor network $(G, \mathcal{M})$ can be contracted in time $n2^{\mathrm{vc}(G)+1}$ and space $n2^{\mathrm{vc}(G)+1}$, or in time $2^{1.5\mathrm{ec}(G)+1}$ and space $2^{\mathrm{ec}(G)+1}$. More precisely, the tensor network can be contracted in time $\min_{(T,b)} \sum_{t \in T} 2^{\mathrm{vc}(t)}$, where the minimization is over contraction trees (T, b). The contraction can be done using space equal to the minimum weighted, directed modified cutwidth of a rooted contraction tree using edge weights $w(f) = 2^{\mathrm{ec}(f)}$. If the contraction is done as a series of matrix multiplications, these precise space and time bounds are tight (though not necessarily simultaneously achievable).*

▶ **Theorem 2.** *A parallel algorithm can contract a tensor network $(G, \mathcal{M})$ in time $\min_{(T,b)} \max_l \sum_t 2^{\mathrm{vc}(t)}$, where the minimization is over rooted contraction trees (T, b), the maximization is over leaves l of T, and the summation is over vertices of t on the unique path from the leaf l to the root r. In other words, the time is the minimum vertex-weighted height of a rooted contraction tree, where the weight of a vertex is $w(t) = 2^{\mathrm{vc}(t)}$. If the contraction is done as matrix multiplications in parallel, this is tight.*

Given a tree decomposition of a line graph with width $k - 1$, we can efficiently construct a contraction tree of the original graph with vertex congestion k. Thus one immediate application of our framework is as a way of precisely assessing the costs of contraction implied by different tree decompositions (even of the same width) computed using existing methods. This is especially useful in distinguishing between contraction orders that have the same time requirements but different space requirements; prior to this work, there was no comprehensive way of quantifying the space requirements, which in practice can be the limiting factor. Alternatively, one can start with existing algorithms for computing good branch decompositions, which can be converted into contraction trees of small edge congestion. More broadly, identifying the right abstraction (i.e., contraction trees) and precise quantification of the space and time costs is a foundation for minimizing those costs as much as possible.

In Section 2, we go over the graph-theoretic concepts that are the foundation of this work. In Section 3, we present seemingly unrelated graph properties in a unified framework that may be of independent interest. Section 3, while strictly unnecessary for understanding the

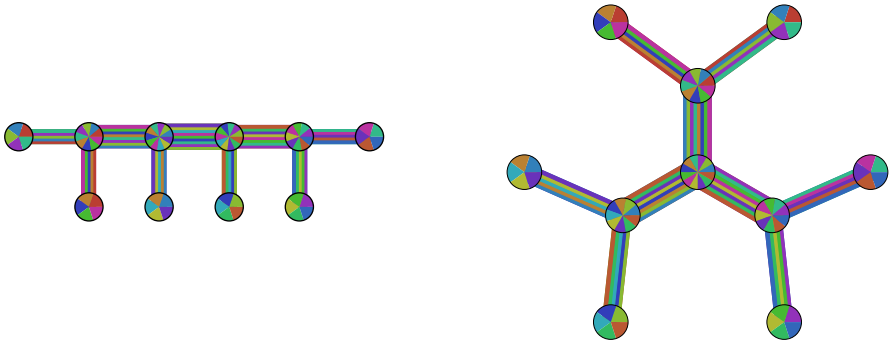


Figure 1 Two unrooted contraction trees for a tensor network with 6 tensors. Each color corresponds to a wire of the tensor network. The inclusion of a color in the representation of a vertex or edge of the contraction tree indicates the contribution of the corresponding wire's weight to the congestion of the vertex or edge, respectively.

main results, helps explain the relationship between our work and prior work. In Section 4, we introduce the cost model on which our results are based. In Section 5, we give our main results. In Section 6, we discuss extensions and generalizations of the main ideas. In Section 7, we conclude with some possible directions for future work. In Appendix A, we prove that the edge congestion of a graph is almost equal to the branchwidth of its line graph.

2 Background

Let $[i,n] = \{j \in \mathbb{Z} | i \leq j \leq n\}$, $[n] = [1,n]$, and $[\mathbf{n}] = [n_1] \times [n_2] \times \cdots \times [n_r]$ for $\mathbf{n} = (n_1, \ldots, n_r) \in (\mathbb{Z}^+)^r$. Let $G[S] = \left(V, E \cap \binom{S}{2}\right)$ be the subgraph of G induced by a subset of the vertices $S \subset V(G)$. For two disjoint sets of vertices of an edge-weighted graph, $w(S,S') = \sum_{\{u,v\} \in E | u \in S, v \in S'} w(\{u,v\})$ is the sum of the weights of the edges between $S \subset V$ and $S' \subset V$. More generally, for r disjoint sets of vertices, $w(S_1, \ldots, S_r) = \sum_{\{i,j\} \in \binom{[r]}{2}} w(S_i, S_j)$ is the sum of the weights of the edges with endpoints in distinct sets. In this context, we will denote singleton sets by their sole arguments, e.g., $w(u,v) = w(\{u\},\{v\}) = w(\{u,v\})$.

2.1 Tensor networks and contraction

A *tensor* can be defined in several equivalent ways. Most concretely, it is a multidimensional array. Specifically, a rank-r tensor is an r-dimensional array of size $\mathbf{d} = (d_1, \ldots, d_r)$. More abstractly, a tensor is a collection of numbers indexed by the Cartesian product of some set of indices, e.g., $[T_{i_1,i_2\ldots,i_r}]_{(i_1,i_2,\ldots,i_r) \in [d_1] \times [d_2] \times \cdots \times [d_r]}$ indexed by $\mathbf{i} \in [\mathbf{n}]$. Alternatively, a tensor can be thought of as a multilinear map $T : [\mathbf{d}] \to \mathbb{C}$. (Our focus will be on complex-valued tensors.)

▶ **Definition 3** (Tensor network). *A* tensor network $(G, \mathcal{M})$ *is an undirected graph* G *with edge weights* w *and a set of tensors* $\mathcal{M} = \{\mathbf{M}_v | v \in V(G)\}$ *such that* $\mathbf{M}_v$ *is a* $|N(v)|$*-rank tensor with* $2^{\deg(v)}$ *entries, where* $\deg(v) = \sum_{u \in N(v)} w(\{u,v\})$ *is the weighted degree of* v*. Each edge* e *corresponds to an index* $i \in \left[2^{w(e)}\right]$ *along which the adjacent tensors are to be contracted.*

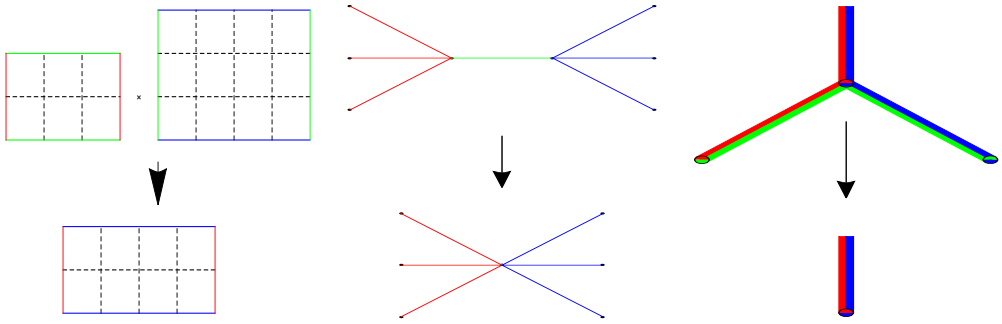


Figure 2 Three ways of viewing the contraction of two tensors. Left: multiplication of a $d_{\mathrm{L}} \times d_{\mathrm{M}}$-rank tensor with a $d_{\mathrm{M}} \times d_{\mathrm{R}}$-rank tensor, resulting in a $d_{\mathrm{L}} \times d_{\mathrm{R}}$-rank tensor. Middle: contraction of a degree $w_{\mathrm{L}} + w_{\mathrm{M}}$ vertex with a degree $w_{\mathrm{M}} + w_{\mathrm{R}}$ vertex, resulting in a degree $w_{\mathrm{L}} + w_{\mathrm{R}}$ vertex. Right: removing a close pair of leaves (with congestions $w_{\mathrm{L}} + w_{\mathrm{M}}$ and $w_{\mathrm{M}} + w_{\mathrm{R}}$) of a contraction tree, leaving a new leaf with congestion $w_{\mathrm{L}} + w_{\mathrm{R}}$.

The contraction of two tensors is the summation over the values of their shared indices. Graphically, this is like an edge contraction of the edge adjacent to the two tensors. The result is a new tensor that takes the place of the two original ones. [1] See Figure 2. Let v_1 and v_2 be the vertices contracted into the new vertex $v_{\{1,2\}}$. The weight of an edge between the new vertex and any other vertex v' is $w\left(v_{\{1,2\}}, v'\right) = w\left(\{v_1, v_2\}, v'\right) = w\left(v_1, v'\right) + w\left(v_2, v'\right)$.

Except in Section 6, we assume that all tensor networks have no "open legs", i.e., every edge connects two vertices (tensors). In this case, the value of a tensor network is the single number that results from contracting all of its edges. Each contraction reduces the number of vertices (tensors) by one, so the network is fully contracted by $n - 1$ contractions. We call a sequence of such contractions a *contraction order*. The value of the tensor network is independent of the contraction order, but the cost of doing the contraction can vary widely depending on the contraction order. Each contraction is identified by an edge, but that edge may not be in the original graph, i.e., its adjacent vertices may have been formed by earlier contractions. One way of specifying a contraction order is by a sequence of edges of the original graph that constitute a spanning tree thereof. In Section 5, we introduce the notion of contraction trees, which allow for a conceptually clear way of expressing contraction orders that makes manifest the associated temporal and spatial costs.

Exactly computing the value of a tensor network is #P-hard [5], as a tensor network can be constructed that counts the number of satisfying assignments to a satisfiability instance or the number of proper colorings of a graph. Even multiplicative and additive approximation is NP-hard [3]. Interestingly, approximating the value of a tensor network with bounded degree and bounded bond dimension is BQP-complete [3]. That is, not only can tensor networks simulate quantum circuits, but quantum circuits can simulate tensor networks as well. In this sense, tensor networks and quantum circuits are computationally equivalent.

[1] Note that this is a "parallel" model of contraction, whereas Markov and Shi use "one-edge-at-a-time" contraction of multigraphs. They are equivalent in the sense that an edge with integer weight can be considered as that number of (unweighted) parallel edges. The parallel model more closely matches how contraction is done in practice. It also allows for arbitrary bond dimension, whereas the multigraph model requires that all bond dimensions be powers of the same base.

2.2 Treewidth and branchwidth

This section is intended primarily to establish notation and recapitulate the standard definitions of the graph properties used in the present work. For a more thorough and pedagogical treatment, see Diestel's excellent textbook [17]. Many instances of graph problems that are hard in general are actually easy when instance graphs are restricted to trees. In many such cases, this generalizes in the sense that it is possible to characterize the hardness of an instance by how "tree-like" it is, as captured by the treewidth of the graph. The treewidth of a graph is defined in terms of an optimal tree decomposition. Treewidth has several alternative characterizations; one of these, *elimination width*, is the basis of Markov and Shi's result equating treewidth and contraction complexity.

▶ **Definition 4** (Tree decomposition). *A* tree decomposition *of a graph $G = (V, E)$ is a tuple $(T, \mathcal{X})$ of a tree T and a tuple $\mathcal{X} = (X_t)_{t \in V(T)}$ of subsets (called* bags*) of the vertices of G with the following properties.*

1. *For every edge $\{u, v\} \in E(G)$, there is some bag $X \in \mathcal{X}$ that contains both endpoints: $u, v \in X$.*
2. *For every vertex $v \in V(G)$ of G, the subtree $T[S_v]$ of T induced by the bags $S_v = \{X \in \mathcal{X} | v \in X\}$ containing v is non-empty and connected.*

▶ **Definition 5** (Width and treewidth). *The* width *of a tree decomposition $(T, \mathcal{X})$ of a graph G is one less than the size of the largest bag:* $\mathrm{width}(G, T, \mathcal{X}) = \mathrm{width}(\mathcal{X}) = \max_{X \in \mathcal{X}} |X| - 1$. *The* treewidth *of a graph is the minimum width of a tree decomposition of the graph.*

A related concept is that of path decompositions and pathwidth, defined analogously to tree decompositions and treewidth, except restricted to paths rather than trees.

▶ **Definition 6** (Path decomposition and pathwidth). *A* path decomposition *of a graph G is a tree decomposition $(T, \mathcal{X})$ of G such that T is a path. The pathwidth* $\mathrm{pathwidth}(G)$ *of G is the minimum width of a path decomposition of G.*

▶ **Definition 7** (Branch decomposition). *A* branch decomposition *of a graph $G = (V, E)$ is a tuple (T, b) of a binary tree T and a bijective function $b : E(G) \to V(T)$ between the edges E of G and the leaves of T.*

For each vertex $v \in V(G)$ of G, let $S_v \subset V(T)$ be the minimal spanning tree of T that contains all the leaves corresponding to edges adjacent to v.

▶ **Definition 8** (Branchwidth). *The* width*, denoted* $\mathrm{width}_G(T, b, \{s, t\})$*, of an edge $\{s, t\} \in E(T)$ of a branch decomposition (T, b) of a graph G is $|\{v \in V(G) | \{s, t\} \subset S_v\}|$, i.e., the number of vertices of G such that the subtree $T[S_v]$ contains $\{e, t\}$. The* width *of the branch decomposition is the largest width of an edge,* $\mathrm{width}_G(T, b) = \max_{f \in E(T)} \mathrm{width}_G(T, b, f)$. *The* branchwidth $\mathrm{branchwidth}(G) = \min_{(T,b)} \mathrm{width}_G(T, b)$ *of a graph is the minimum width of a branch decomposition thereof.*

2.3 Congestion

There is an alternative but less explored way of quantifying how "tree-like" a graph is: the minimum congestion of a tree embedding, introduced by Bienstock [7].[2]

[2] Note that this is entirely distinct from a different type of congestion problem in which the goal is find routings for some specified set of pairs of terminals.

▶ **Definition 9** (Tree embedding). *A* tree embedding *of a graph G is a tuple (T, b) of a binary tree T and a bijection $b : V(G) \to V(T)$ between the vertices of G and the leaves of T.*

Let $S_{v,w}$ be the unique path between the leaves $b(v)$ and $b(w)$ of T.

▶ **Definition 10** (Congestion). *The congestion of a vertex $v \in V(T)$ (resp., edge $f \in E(T)$) is the total weight of the edges $e \in E(G)$ whose subtrees S_e include v (resp., f).*

2.4 Cutwidth

▶ **Definition 11** (Cutwidth). *Let $f : V \to [n]$ be a linear ordering of the vertices of a graph $G = (V, E)$. The cutwidth of f is the maximum number of edges that cross a gap:*

$$\max_{i \in [n-1]} |\{\{u, v\} \in E | f(u) \leq i < f(v)\}| .$$

The modified cutwidth of f is the maximum number of edges that cross a vertex:

$$\max_{i \in [n]} |\{\{u, v\} \in E | f(u) < i < f(v)\}| .$$

The cutwidth (resp., modified cutwidth) of a graph is the minimum cutwidth (resp., modified cutwidth) of a linear ordering. For edge weighted graphs, the weighted cutwidth and modified cutwidth count the total weights of the relevant edge sets rather than their cardinalities. For a directed acyclic graph, the directed cutwidth (resp., modified cutwidth) is the minimum cutwidth (resp., modified cutwidth) of a linear ordering that is topologically sorted according to the graph.

2.5 Parameterized complexity

Approximating both treewidth and pathwidth to within a constant factor is NP-hard, though there exist efficient algorithms for logarithmic and polylogarithmic approximations, respectively [10, 11]. However, deciding whether or not the treewidth is at most some constant can be done in linear time (albeit it with an enormous prefactor) [8]. For many graph problems, e.g., Maximum Independent Set, there exist algorithms whose run time is exponential only in the treewidth or pathwidth, i.e., given the instance graph and a tree decomposition thereof of width k, the algorithm runs in time $2^k n^{O(1)}$ [4]. The Exponential Time Hypothesis (ETH) implies that several such parameterized complexity results are optimal, in the sense that there exists no $2^{o(k)} n^{O(1)}$ algorithm [16].

The situation is similar for branchwidth. Computing the branchwidth of a graph is in general NP-hard, but can be done efficiently for planar graphs [26]. (Whether computing the treewidth of a planar graph is NP-hard is an open question.) As is the case for treewidth, there is a constructive linear time algorithm for deciding whether or not the branchwidth is at most some constant (and in this case with better constant factors) [12]. Good branch decompositions can be used to implement dynamic programming algorithms for problems such as the traveling salesman problem [15].

Computing the vertex congestion of a graph is claimed to be NP-hard [7], but no proof appears in the literature.

Computing the (edge) cutwidth is NP-hard, but for any constant k, a linear ordering of cutwidth k (for all variants) can be found in linear time if one exists [9].

Leaves	Subtrees	Minimization over	Target family: Trees	Target family: Caterpillars
Edges	Vertices	Vertices	Treewidth	Pathwidth
Edges	Vertices	Edges	Branchwidth	
Vertices	Edges	Vertices	Vertex congestion	Modified cutwidth
Vertices	Edges	Edges	Edge congestion	

Figure 3 Table of graph properties. Each row corresponds to an instantiation of Equation 1.

3 Unified framework of graph properties

In this section, we present a unified framework of various graph properties, as captured in the following combined definition:

A $\left\{\begin{smallmatrix}\text{tree decomposition}\\ \text{branch decomposition}\\ \text{tree embedding}\\ \text{tree embedding}\end{smallmatrix}\right\}$ of a $\left\{\begin{smallmatrix}\text{vertex}\\ \text{vertex}\\ \text{edge}\\ \text{edge}\end{smallmatrix}\right\}$-weighted graph G is a tuple (T, b) of a binary tree T and a bijection b between the leaves of T and the $\left\{\begin{smallmatrix}\text{edges}\\ \text{edges}\\ \text{vertices}\\ \text{vertices}\end{smallmatrix}\right\}$ of G. The bijection b implies a subtree for every $\left\{\begin{smallmatrix}\text{vertices}\\ \text{vertices}\\ \text{edges}\\ \text{edges}\end{smallmatrix}\right\}$ of the graph. The $\left\{\begin{smallmatrix}\text{treewidth}\\ \text{branchwidth}\\ \text{vertex congestion}\\ \text{edge congestion}\end{smallmatrix}\right\}$ of the graph is the minimum over all $\left\{\begin{smallmatrix}\text{tree decompositions}\\ \text{branch decompositions}\\ \text{tree embeddings}\\ \text{tree embeddings}\end{smallmatrix}\right\}$ of the maximum total weight of all subtrees containing any $\left\{\begin{smallmatrix}\text{vertex}\\ \text{edge}\\ \text{vertex}\\ \text{edge}\end{smallmatrix}\right\}$. The $\left\{\begin{smallmatrix}\text{pathwidth}\\ \text{—}\\ \text{modified cutwidth}\\ \text{—}\end{smallmatrix}\right\}$ is defined in the same way as $\left\{\begin{smallmatrix}\text{treewidth}\\ \text{—}\\ \text{vertex congestion}\\ \text{—}\end{smallmatrix}\right\}$ except that T is restricted to be a caterpillar. (1)

Let's unpack this. For branchwidth and congestions, (1) is the standard definition. For the others, (1) is non-standard but equivalent to the standard definitions. Writing them all in this way helps elucidate the relationships between them, which are obscured by the standard definitions.

Note that both the vertex and edge congestions of a graph G are defined as optimal properties of the same type of object, namely a tree embedding (T, b). For every edge $e \in E(G)$, the mapping $b : V(G) \to V(T)$ of the vertices to leaves of the tree implies a minimal subtree S_e connecting the leaves of T corresponding to its endpoints in G. (For an edge of size 2, this subtree S_e is a path, but the definition allows for hyperedges as well.) The vertex and edge congestions are then the maximum total weight of subtrees that contain any vertex or edge, respectively, of the tree T.

There is a similar relationship between treewidth and branchwidth. Usually, we think of a tree decomposition of a graph $G = (V, E)$ as a tree T and a subtree S_v for every vertex in $V(G)$ such that the subtrees for every pair of adjacent (in G) vertices overlap. In (1), S_v is specified implicitly as the (unique) minimal spanning subtree of T that connects the leaves of T corresponding to the edges of G that are incident to v. By design, this tree T and set of subtrees is the same as that for a branch decomposition. The treewidth and branchwidth are the maximum total weight of subtrees (now corresponding to vertices of G) that contain any vertex or edge, respectively, of the tree T.

So we see that the congestions are defined by the overlap of subtrees of T corresponding to edges of G and that the tree- and branchwidths are defined by the overlap of subtrees of T corresponding to vertices of G, the former implied by a mapping from vertices of G to leaves

of T and the latter by a mapping from edges of G to leaves of T. The vertex congestion and treewidth are concerned with the overlap at vertices of T, and the edge congestion and branchwidth with the overlap on edges of T. Thus we have made the analogy that treewidth : branchwidth :: (vertex congestion) : (edge congestion). For example, that [25] $\mathrm{bw}(G) \leq \mathrm{tw}(G) \leq \frac{3}{2}\mathrm{bw}(G)$ and [7] $\mathrm{ec}(G) \leq \mathrm{vc}(G) \leq \frac{3}{2}\mathrm{ec}(G)$ is no coincidence.

Now consider the line graph $L(G) = (E, \{\{e, e'\} \subset E | e \cap e' \neq \emptyset\})$ of a graph $G = (V, E)$. Suppose we have a tree embedding (T, b) of the original graph G, with an implied subtree T_e for every edge $e \in E(G)$. Because the vertices of the line graph $L(G)$ correspond to the edges of G, this can be considered as a branch decomposition of the line graph $L(G)$. For every pair of edges $e, e' \in E(G) = V(L(G))$ that are adjacent in the line graph, the corresponding subtrees $S_e, S_{e'}$ intersect at the leaf $b(v) \in V(T)$, where $v \in e, e'$ is the vertex of G adjacent to e and e'. The vertex congestion of the tree embedding (T, b) is the width of (T, b) interpreted as a tree decomposition, and the edge congestion of the tree embedding is the width of (T, b) interpreted as a branch decomposition. This implies that $\mathrm{tw}(L(G)) \leq \mathrm{vc}(G)$ and $\mathrm{bw}(L(G)) \leq \mathrm{ec}(G)$. Actually, these inequalities are tight or almost so: $\mathrm{tw}(L(G)) = \mathrm{vc}(G)$ and $\mathrm{bw}(L(G)) \leq \mathrm{ec}(G) \leq \mathrm{bw}(L(G)) + \left\lfloor \frac{\deg(G)}{3} \right\rfloor$. The other direction, going from a tree decomposition to a tree embedding, requires seeing that a tree decomposition of a line graph can be made to have a particular structure, specifically that the edges of $L(G)$ corresponding to each vertex of G can be mapped to disjoint subtrees of T. The equality was shown by Harvey and Wood [21] and captures how our characterization of the temporal costs of tensor network contraction relates to earlier characterizations. However, our characterization in terms of tree embeddings, while mathematically equivalent to that in terms of tree decompositions of line graphs, allows for a conceptually cleaner and more fine-grained perspective. We prove the inequalities in Appendix A.

▶ **Theorem 12.** *The edge congestion of graph G is at least the branchwidth of its line graph and at most the same plus a third of its maximum degree. Furthermore, a tree embedding with edge congestion $k + \lfloor \deg(G)/3 \rfloor$ can be efficiently computed from a branch decomposition of width k and a branch decomposition of width k can be efficiently computed from a tree embedding with edge congestion k.*

For vertex congestion and treewidth (which concern the overlap of subtrees at vertices), the requirement that the mapping be a bijection with the leaves of the tree can be dropped, as can the requirement that the tree be binary. Yet these requirements are without loss of generality, as any tree embedding or tree decomposition can be modified to satisfy these without increasing its vertex congestion or treewidth, respectively. For edge congestion and branchwidth, which concern overlap over edges, the bijection and degree requirements are essential.

The usual definitions for pathwidth and modified cutwidth are in terms of paths (or, equivalently, linear orderings), whereas in (1) we allowed them to be caterpillars. This is equivalent, and allows us to relate the properties just discussed with their linear variants. In particular, the relationship between the bubblewidth of a tensor network $(G, \mathcal{M})$ and its "contraction complexity" is almost the same as that between the modified cutwidth of the graph and its vertex congestion, in the sense that the bubblewidth is exactly equal to the cutwidth and $\mathrm{cw}(G) \leq \mathrm{mcw}(G) \leq \mathrm{cw}(G) + \deg(G)$.

We can make another analogy, that treewidth : (vertex congestion) :: pathwidth : (modified cutwidth) . For example, [9, 21] $\frac{1}{2}(\mathrm{tw}(G) + 1) \leq \mathrm{vc}(G) + 1 \leq \deg(G)(\mathrm{tw}(G) + 1)$ and $\mathrm{pw}(G) \leq \mathrm{mcw}(G) + 1 \leq \deg(G)(\mathrm{pw}(G) + 1)$.

The (unmodified) cutwidth is a linear analog to what Ostrovskii called the "tree congestion" of a graph [23]; the tree congestion is the same as the edge congestion except that there is a bijection between *all* the vertices of the binary tree, rather than just the leaves.

4 Contraction costs

Our primary motivation is minimizing the time and space costs of tensor network contraction. Ideally, for instances of interest we would like to *provably* minimize the cost, which entails tight lower bounds and the corresponding constructions that meet them. Given the formal hardness of tensor network contraction and the informal hardness of proving lower bounds, we restrict our attention to minimizing the cost of tensor network contraction as it is most commonly done: as a series of matrix multiplications.

First, how much space is required to store a tensor network $(G, \mathcal{M})$? Each tensor $\mathbf{M}_v$ consists of $2^{\deg_v}$ numbers; this is the main component of the space requirements. Technically, we must also keep track of the graph G and the weights of its edges $E(G)$ as well as a dope vector for each tensor indicating how the tensor is laid out in memory; these will be negligible. Our memory accounting will be in units of whatever is used to store a single entry of a tensor. While in general, the bit depth of an entry may scale non-trivially with instance size, practical implementations will use a fixed-width data type.

Then, what do we need to do a contraction of two tensors? Suppose we want to contract a $(d_\mathrm{L}, d_\mathrm{M})$ tensor with a $(d_\mathrm{M}, d_\mathrm{R})$ tensor along their shared dimension d_M. The input tensors require a total of $d_\mathrm{M}(d_\mathrm{L} + d_\mathrm{R})$ space and the output tensor $d_\mathrm{L} d_\mathrm{R}$. In theory, it should be possible to do the contraction using no more space than that required by the larger of the input tensors and output tensor. In practice, new memory is allocated for the new tensor, it is populated with the appropriate data from the input tensors, and then the memory for the latter is freed. We assume the second cost model, in which memory is simultaneously allocated both for the tensors to be contracted and for the tensor that results from their contraction, but our ideas are straightforwardly modifiable for plausible variants.

The contraction itself is essentially matrix multiplication, and a straightforward implementation will take time $d_\mathrm{L} \cdot d_\mathrm{M} \cdot d_\mathrm{R}$. There exist Strassen-like algorithms for matrix multiplication with better asymptotic runtime, but the constant pre-factors are so large and the straightforward algorithm so heavily optimized that they are of little practical value given the size of currently available machines.

Lastly, in order to implement a tensor contraction as a matrix multiplication, the tensors must be laid out commensurately in memory. If they are not, then the data of one tensor or both must be permuted to make them so. This permutation can effectively be done in place and in linear time. In practice, the permutation time is negligible compared to the matrix multiplication time.

5 Contraction orders and trees

In this section, we present our main contribution: a graph-theoretic characterization of the temporal and spatial costs of families of contraction orders.

5.1 Linear contraction orders

We start with a special case of contraction orders. Let a *linear contraction order* be one specified by an ordering of the vertices $(v_1, v_2, \ldots, v_n)$. That is, the first contraction is of vertices v_1 and v_2 to form a new vertex $v_{1,2}$. The second contraction is of $v_{1,2}$ and v_3 to form $v_{1,2,3}$, and so on. We represent such a contraction order by what we call a *rooted contraction tree*. The contraction tree of a linear contraction order is a binary caterpillar tree with $n+1$ leaves, one for each vertex of the original graph and a special leaf called the *root*, as shown in Figure 4. The root leaf is at one of the "ends" of the tree. Each vertex v_i for $i \in [2, n]$ is at

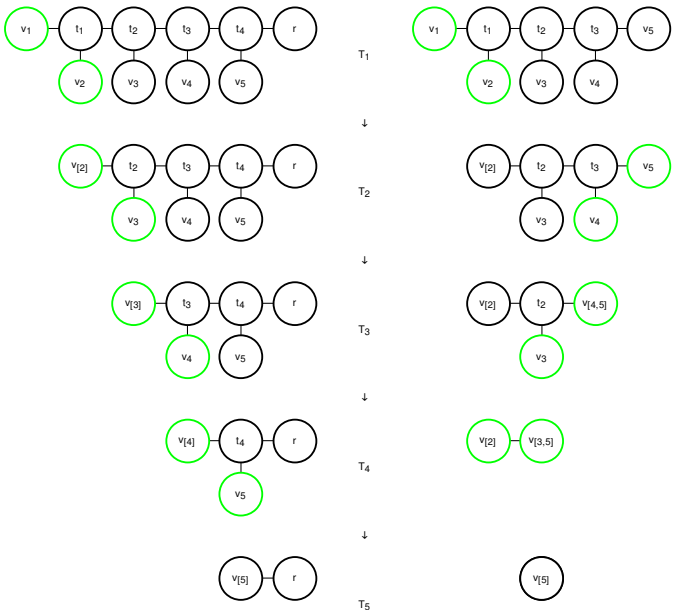


Figure 4 Two series of contraction trees (unrooted and rooted on left and right, respectively) for a tensor network with 5 tensors. From top to bottom, the contraction trees for the initial, intermediate, and final tensor networks. The pair of leaves corresponding to the next pair of tensors to be contracted are highlighted in green.

distance $n+2-i$ from the root, and vertex v_1 is at distance n therefrom. We denote such a contraction tree by (T, b), where T is the tree and $b : V(G) \to V(T) \cup \{r\}$ is the bijection between the vertices of G and the leaves of T together with the root r.

Recall that for a tensor network $(G, \mathcal{M})$, we are using the convention that the weight $w(u, v)$ of an edge $\{u, v\}$ is the logarithm of the bond dimension of wire connecting tensors $\mathbf{M}_u$ and $\mathbf{M}_v$. For each edge $\{u, v\}$ of G there is a unique path in T between $b(u)$ and $b(v)$, which we call a *routing*. Assign the weight $w(u, v)$ to every vertex and edge on this path, including the endpoints $b(u)$ and $b(v)$. We say that the *congestion* of a vertex or edge of T, denoted $\mathrm{con}(v)$ or $\mathrm{con}(e)$, is the sum of the weights of all the routings that include it. Label the non-root leaves of T by $l_i = b(v_i)$ and the internal vertices by t_i for $i \in [n-1]$, where t_{n-1} is closest to the root and t_1 is farthest. For concision, identify t_0 with t_1.

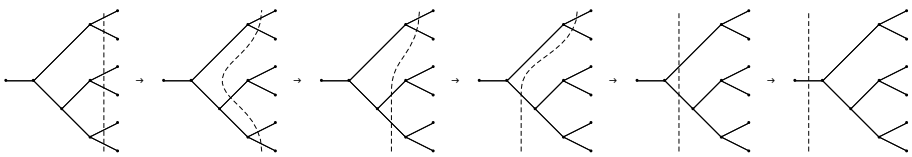


Figure 5 The intermediate states of a contraction procedure. The tree pictured is a rooted contraction tree, with the root at the left. The dashed line crosses edges of the contraction tree adjacent to tensors held in memory.

We now show that these congestions capture the costs of the contraction order. First, note that for each vertex $v \in G$, the congestion $\mathrm{con}(e)$ of the edge $e \in E(T)$ adjacent to $b(v)$ gives the size of the tensor $\mathbf{M}_v$, in the sense that $\mathrm{con}(e) = \sum_{u \in V(G)} w(v,u) = \deg_v$, so that $2^{\mathrm{con}(e)}$ is the product of the bond dimensions of the tensor M_v. Now, consider the first contraction, of vertices v_1 and v_2, i.e., tensors $\mathbf{M}_{v_1}$ and $\mathbf{M}_{v_2}$. The bond dimension of the wire between them is $2^{w(v_1,v_2)}$. The product of the bond dimensions of $\mathbf{M}_{v_1}$ with tensors besides v_2 is $2^{\deg_{v_1} - w(u,v)}$, and similarly for $\mathbf{M}_{v_2}$. As discussed in Section 4, the contraction can be done in time $2^{\deg_{v_1} - w(u,v)} \cdot 2^{w(u,v)} \cdot 2^{\deg_{v_2} - w(v_1,v_2)} = 2^{w(v_1,v_2,V(G)\setminus\{v_1,v_2\})}$, where $w(v_1, v_2, V(G) \setminus \{v_1, v_2\})$ is the total weight of edges across the tripartite cut. This is exactly the congestion of the vertex $t \in V(T)$ adjacent to both $b(v_1)$ and $b(v_2)$. Suppose that we have done the contraction, yielding a new tensor network containing the contracted vertex $v_{1,2}$. The size of this new tensor $\mathbf{M}_{v_{1,2}}$ is $2^{w(\{v_1,v_2\},V\setminus\{v_1,v_2\})} = 2^{\mathrm{con}(\{t_1,t_2\})}$. If we continue with the contractions, we notice an exciting pattern. We can identify each contraction with an internal vertex of T. The congestion of that vertex gives the time to do the contraction, and the congestion of the adjacent edge nearest the root gives the space of the resulting contracted tensor. The congestion of the leaves, which is equal to the congestions of the adjacent edges and gives the size of the corresponding tensors, can be interpreted as giving the time required to simply read in the tensors of the initial network to be contracted. Overall, the total time of all the contractions is $\sum_{t \in V(T)} 2^{\mathrm{con}(t)} \leq 2n \cdot 2^{\mathrm{vertcon}_{T,b}(G)}$, where $\mathrm{vertcon}_{T,b}(G) = \max_{t \in V(T)} \mathrm{con}(t)$. Furthermore, each edge $e \in E(T)$ corresponds to a tensor $\mathbf{M}_e$ that appears at some point in the series of contractions; those adjacent to leaves correspond to the initial tensors and internal edges to tensors resulting from contractions. The congestion of each edge gives the size of the corresponding tensor, in the sense that the size of $\mathbf{M}_f$ is $2^{\mathrm{con}(f)}$. At any point point in the contraction order, there are at most n tensors, so the required memory is at most $n2^{\mathrm{edgecon}_{T,b}(G)}$, where $\mathrm{edgecon}_{T,b}(G) = \max_{f \in E(T)} \mathrm{con}(f)$. As shown in Section 3, the minimum vertex congestion over all linear contraction orders is exactly equal the vertex cutwidth of G. It is closely related to the bubblewidth of earlier work [3], which is exactly equal to the edge cutwidth. The minimum edge congestion over all linear contraction orders is exactly equal to the edge cutwidth of G.

5.2 General contraction orders

We now turn our attention to general (i.e., not necessarily linear) contraction orders. The first generalization we make is to remove the root. In other words, for each linear contraction order we form an *unrooted contraction tree* exactly as before except that leaves of T are in unqualified bijection with the vertices of G. This unrooted contraction tree can be interpreted as corresponding to 2^{n-2} different contraction orders in the following way. Define a pair of leaves in a binary tree to be *close* if they are at distance 2. In the caterpillar binary trees we have considered thus far, there are two pairs of close leaves, at each "end" of the tree. Before, we used a rooted caterpillar contraction tree to represent the unique contraction order given by contracting the two non-root close leaves until we got to the root. Now, the unrooted caterpillar contraction tree represents the family of contraction orders that can be specified by contracting *either* pair of a close leaves of the contraction tree until a single vertex remains. Importantly, it remains true that every one of these contraction orders takes time exactly $\sum_{t \in V(T)} 2^{\mathrm{con}(t)} = \tilde{O}\left(2^{\mathrm{vertcon}_{T,b}(G)}\right)$ and space $\tilde{O}\left(2^{\mathrm{edgecon}_{T,b}(G)}\right)$.

The second generalization we make is to remove the restriction to caterpillar trees.

▶ **Definition 13.** *A* rooted contraction tree (T, b) *of a tensor network* $(G, \mathcal{M})$ *is a rooted binary tree* T *and a bijection* $b : V(G) \to V(T)$ *between the vertices (tensors) of* G *and the (non-root) leaves of* T*. An* unrooted contraction tree (T, b) *is an unrooted binary tree* T *and a bijection* $b : V(G) \to V(T)$ *between the vertices of* G *and the leaves of* T*.*

An unrooted contraction tree represents a set of contraction orders in the following way. Suppose we have a contraction order $e_1, \ldots, e_{n-1}$; Each edge can be written as $e_i = \{v_S, v_{S'}\}$ for some disjoint $S, S' \subset V(G)$, where v_S is the vertex formed by contracting the vertices in S. We start with an empty forest $T_1 = (V(G), \emptyset)$. For each contraction $e_i = \{v_S, v_{S'}\}$, we add a new vertex $v_{S \cup S'}$ to the forest, as well as edges from the new vertex to v_S and $v_{S'}$. That is, $T_i = (V(T_{i-1}) \cup \{v_{S \cup S'}\}, E(T_{i-1}) \cup \{\{v_S, v_{S \cup S'}\}, \{v_{S'}, v_{S \cup S'}\}\})$. For the last contraction, instead of adding a new vertex, we only add an edge between v_S and $v_{S'}$. Doing this yields an unrooted contraction tree for the given contraction order. We say that an unrooted contraction tree represents the set of contraction orders from which it can be constructed in this way. If for the last contraction, we added not only a new vertex v_V connected to v_S and $v_{S'}$ but a second vertex r connected v_V, we would have a rooted contraction tree.

We can easily go the other way as well. Suppose we have a rooted contraction tree. Then we can iteratively build a contraction order. We select an arbitrary close pair of non-root leaves, and add to the contraction order the contraction corresponding to the adjacent internal vertex (of the tree). We then remove the two leaves and their adjacent edges. The internal vertex now becomes a leaf, and corresponds to the vertex resulting from contracting the two vertices (of the tensor network) in the new contraction tree. We repeat until only a single edge of the tree remains, corresponding to the completely contracted tensor network and the root. This is the same procedure visualized in Figure 4, except that, when the contraction tree is not restricted to be a caterpillar, there may be many more than two pairs of close leaves to choose from at each step.

Given an unrooted contraction tree, we can turn it into a rooted contraction tree by splitting any edge (i.e., removing an edge, adding a new vertex and adding edges between the new vertex and the vertices adjacent to the removed one), and then adding a root vertex and connecting it to the first newly inserted vertex.

Proof of Theorem 1. In a contraction tree, either rooted or unrooted, each internal vertex corresponds to a contraction. In rooted contraction trees, there is a clear directionality; two of the neighbors are "inputs" and the third is "output". However, the congestion of the vertex, the exponential of which gives the time to do the matrix multiplication, is independent of this directionality. Similarly, each edge of a contraction tree corresponds to a tensor that exists at some point in the contraction (specifically, when the edge is adjacent to a leaf). Again the congestion of this edge is independent of its direction, and the size of the tensor is the exponential of the congestion. Without loss of generality, we prove the theorem using rooted contraction trees.

Suppose we have a rooted contraction tree (T, b) of a tensor network $(G, \mathcal{M})$. Each internal vertex $i \in V(T)$ corresponds to a matrix multiplication, which takes time $2^{\mathrm{vc}(i)}$. Each leaf $l \in V(T)$ corresponds to an initial tensor of size $2^{\mathrm{vc}(l)}$, where $\mathrm{vc}(l) = \deg_G\left(b^{-1}(l)\right)$. Overall, the total time then is $\sum_{t \in V(T)} 2^{\mathrm{vc}(t)} \leq 2n2^{\mathrm{vc}(T)}$.

The rooted contraction tree gives a partial ordering of its vertices, which represent contractions (or initial tensors). Any topologically sorted linear ordering $(t_1, t_2, \ldots, t_{2n-2})$ of the vertices of the contraction tree can be considered uniquely as a contraction order consistent with the contraction tree, and vice versa. For a given contraction order, consider the intermediate state at some point in the overall contraction procedure. Let t_i be the last tensor contracted and t_{i+1} the next one to contract. Each edge $f \in E(T)$ from $\{t_1, \ldots, t_i\}$ to $\{t_{i+1}, \ldots, t_{2n-2}\}$ corresponds to a tensor that needs to be stored at this point. The size of the tensor is exactly $2^{\mathrm{ec}(f)}$. The size of the next tensor (resulting from the contraction corresponding to t_{i+1}) is $2^{\mathrm{ec}(f')}$, where f' is the edge from t_{i+1} towards the root of T. Using the convention that the weight of an edge $f \in E(T)$ of T is $w(f) = 2^{\mathrm{ec}(f)}$, then the directed,

weighted modified cutwidth of a vertex t_{i+1} in a linear ordering $(t_1, \ldots, t_{2n-2})$ of the vertices of T is exactly equal to the space needed to store the remaining tensors to be contracted and make room for the tensor resulting from the next contraction. Once the contraction is done, the memory allocated for the two tensors that were contracted can be freed. For the coarser space bound, we can just pre-allocate memory for every tensor that will arise during the procedure, in total space $\sum_{f \in E(T)} 2^{\mathrm{ec}(f)} \leq 2n2^{\mathrm{ec}(T)}$.

Overall, if we choose a contraction tree T with minimum vertex congestion, i.e., $\mathrm{vc}(G) = \mathrm{vc}(T) \geq \mathrm{ec}(T)$, we get time at most $n2^{\mathrm{vc}(G)}$ and space at most $n2^{\mathrm{vc}(G)}$. If instead we choose a contraction tree T with minimum edge congestion, i.e., $\mathrm{ec}(G) = \mathrm{ec}(T) \geq (2/3)\mathrm{vc}(T)$, we get time at most $n2^{1.5\mathrm{ec}(G)+1}$ and space at most $n2^{\mathrm{ec}(G)}$. Tightness follows from the fact that for any contraction order, we can construct a rooted contraction tree whose properties give the stated bounds. ◀

Proof of Theorem 2. Suppose we have a rooted contraction tree (T, b) and that $l^* \in V(T)$ is a leaf on a longest path from a leaf to the root using the vertex weight $w(t) = 2^{\mathrm{vc}(t)}$. Call this path from l^* to the root the *critical path* $P*$. The vertices on $P*$, ordered from the leaf to the root, represent a series of contractions. This series of contractions can be done in time $\sum_{t \in V(P^*)} 2^{\mathrm{vc}(t)}$, the vertex-weighted length of the P^*, which by definition is the longest such path. We prove the claim for general contraction trees by induction. The base case is a tensor network of just two tensors, so that there is just a single contraction and the critical path has 3 vertices. The inductive step is that if the claim is true for a contraction tree whose critical path has k vertices, it is true for a contraction tree whose critical path has $k + 1$ vertices. Consider the last vertex t on P^* nearest the root. It corresponds to a contraction of a tensor from an earlier contraction $P*$ and a tensor from the remaining subtree of T, i.e., the part of tree not containing P^*. By definition, the length of the critical path of this subtree is no more than the length of the subpath from l^* to t; otherwise P^* would not be the longest path. Therefore, this subtree can be contracted in less time than the earlier parts of P^*. These can be done in parallel, so the overall time is simply that for P^*. ◀

As shown in Appendix A, a branch decomposition of $L(G)$ with width k can be efficiently converted into a contraction tree of G with edge congestion $k + \lfloor \deg(G)/3 \rfloor$. Similarly, a tree decomposition of $L(G)$ with width $k - 1$ can be efficiently converted into a contraction tree of G with vertex congestion k [21]. Thus one way of utilizing these results is to use an existing algorithm for finding tree decompositions or branch decompositions as a starting point. Theorems 1 and 2 can then be used to construct minimum-cost contraction orders in a more precise way than previous results allow. Developing empirically good implementations of algorithms for finding tree decompositions is a particularly active area research [2]. These are already exploited in much recent work on tensor network contraction [13, 14, 27]. The framework presented here can significantly augment the effectiveness of such techniques. For instances with a lot of structure, as typical ones do, the intuitiveness of contraction trees also empowers manual construction of contraction trees.

There are also techniques for certifying the optimality of tree decompositions and branch decompositions (namely, brambles and tangles) that can be ported to certify the optimality of contraction trees with respect to vertex and edge congestion, respectively. For planar graphs, the exact edge congestion can be computed (non-constructively) in polynomial time [26]. In addition to serving as a lower bound for calculations, the structure of such obstructions may help with understanding the complexity of quantum states as represented by tensor networks.

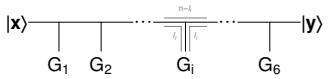


Figure 6 Contraction tree representation of the Schrödinger algorithm for computing $\langle x|G_1G_2\cdots G_m|y\rangle$.

6 Extensions and generalizations

Heretofore, we have assumed that all tensor networks under consideration had no open legs, i.e., that they contract to a single number (0-rank tensor). More generally, we can consider tensor networks with open legs that contract to non-trivial tensors. For such tensors, we treat any open legs as wires to a single "environment" tensor, which we then identify with the root of a rooted contraction tree. For the purposes of minimizing the congestion, the graph will simply have one more vertex. All previous results regarding the costs of contraction then follow exactly as before without modification.

We can also allow tensor networks $(G, \mathcal{M})$ in which G is a hypergraph. Recall how we defined the congestions of a contraction tree (T, b). Each vertex $v \in V(G)$ was identified with a leaf of T through the bijection b. Then each edge $\{u, v\} \in E(G)$ contributed its weight to the congestions of the vertices and edges on the routing (unique path) between $b(u)$ and $b(v)$ in T. For a hyperedge $\{v_1, \ldots, v_k\}$, there is a unique subtree of T connecting the adjacent vertices (which is equal to the union of the paths connecting each pair of edges). Then the hyperedge contributes its weight to the congestions of the vertices and edges on this subtree. The hyperedge corresponds to a so-called "copy" tensor with k legs of the same bond dimension b [5]. The copy tensor is one when all indices have the same value and is zero otherwise. Such a tensor arises, e.g., in a decomposition of a controlled quantum gate.

Decompositions of tensors highlight the main limitation of the present work. While our upper bounds are unconditional, our lower bounds hold only within what we call the matrix multiplication model, in which the only operations allowed are matrix multiplications. This takes advantage only of the topological properties of G and, importantly, not of the properties of the tensor $\mathcal{M}$. However, in many cases of practical interest, the tensors have structure that can be exploited. For example, a tensor corresponding to a quantum gate can be split into two tensors connected by a wire with bond dimension equal to the Schmidt rank across some bipartition of the qubits on which it acts. For gates with less-than-full Schmidt rank, this can help with contraction significantly. Once such a decomposition is made, the sparser graph structure can be exploited by the methods presented here.

Tree-based methods for tensor network contraction are used in state-of-the-art simulations of quantum circuits, where "simulation" here means calculation a single matrix element $\langle \mathbf{x}|C|\mathbf{y}\rangle$ for a pair of basis states $(|\mathbf{x}\rangle, |\mathbf{y}\rangle)$ and the circuit C. In addition to providing a precise analysis of such methods, we can also analyze algorithms not usually expressed in such terms. For example, consider the "Schrödinger" algorithm: a state vector of size 2^n is kept in memory and for each of m gates in sequence. Suppose each gate acts on at most l qubits. Let the circuit be represented as a tensor network with $m + 2$ tensors: one for each gate, one for the output $|\mathbf{x}\rangle$, and one for the input $|\mathbf{y}\rangle$. In the corresponding graph G, the vertex $|\mathbf{x}\rangle$ is adjacent to each of the gate vertices that first act on a qubit, with weight equal to the number of qubits that are first acted on by the gate. Similarly for $|\mathbf{y}\rangle$. The Schrödinger algorithm is then a linear contraction order using the vertex ordering $(|\mathbf{x}\rangle, G_1, \ldots, G_m, |\mathbf{y}\rangle)$. Each internal vertex of the contraction tree adjacent to the vertex corresponding to the l_i-local gate G_i has congestion $n + l_i$: $n - l_i$ from the qubits not acted on by the gate,

then l_i each from the input and output wires. The total time for the contraction is thus $\sum_{i=1}^{m} 2^{n+l_i} \leq m2^{n+l} = O(m2^n)$, where $l = O(1)$ is the maximum locality of a gate. Each internal edge has congestion n, so the contraction can be done using space $O(2^n)$.

An alternative approach is the "Feynman", or path integral, algorithm, which inserts resolutions of the identity after every gate and sums. Now we consider the tensor network corresponding to $\langle \mathbf{x}|C|\mathbf{y}\rangle$ slightly differently. For simplicity, assume all gates are 2-local. Instead of having a single vertex $|x\rangle$ for the input, we have n vertices $|x_i\rangle$, one for each qubit. Similarly, we have n output vertices $\{|y_i\rangle\}$. First, we contract the input vertices $|x_i\rangle$ into the adjacent gate vertices. This leaves $2m$ wires, 2 from each gate to the next or an output. Suppose that instead of contracting the entire tensor network, we remove a single wire and replace it with $|b\rangle\langle b|$ for $b \in \{0,1\}$. The value of the original network is the sum of the values of the reduced networks over $b \in \{0,1\}$. The Feynman algorithm is then to do this for all wires. For each value $\mathbf{b} \in \{0,1\}^{2m}$, we have a tensor network of m tensors and no wires, which we can "contract" in $O(m)$ time and $O(1)$ space. But we need to do this for every $\mathbf{b}$ and sum them up, meaning overall it takes $O(m4^m)$ time. We need $O(n+m)$ space to keep track of $\mathbf{x}$, $\mathbf{y}$, and $\mathbf{b}$. We can generalize this approach to arbitrary tensor networks. First, we remove some set $S \subset E(G)$ of edges, with total weight $W = \sum_{s\in S} w(e)$. There are 2^W values of the corresponding wires, and for each one we contract the reduced tensor network. Let $\tilde{G} = (V, E \setminus S)$ be the reduced network. Overall, for a sequential algorithm, this takes time $\tilde{O}\left(2^{W+\mathrm{vc}(\tilde{G})}\right)$ and space $\tilde{O}\left(W + m2^{\mathrm{ec}(\tilde{G})}\right)$. Moreover, we consider the cuts S as allowing trivial parallelization, by doing the 2^W contractions of the reduced network in parallel on the same number of processors. This idea was used, for example, by Villalonga et al. to balance time and memory usage in their simulation of grid-based random quantum circuits. Aaronson and Chen [1] show that for carefully chosen cuts that form nested partitions, the contributions W to the time and space from the cuts can be significantly reduced.

7 Conclusion

We introduced a graph-theoretic framework for precisely quantifying the temporal and spatial costs of tensor network contraction, with the ultimate goal of minimizing these costs. We conclude with several possible directions for future work:

- Proving the hardness of exactly or approximately computing the vertex or edge congestion of a graph, including of special cases like planar graphs.
- Inventing algorithms (that aren't simply disguised treewidth or branchwidth algorithms) for finding small-congestion contraction trees.
- Exploring the space-time trade-off of vertex and edge congestions. They are always within a small multiplicative constant of each other, but can they be exactly minimized simultaneously? If not, what does the trade-off look like, particularly for graphs of practical interest like 2D and 3D grids.
- Parallelizing at larger scale. In our discussion of parallelized algorithms, we neglected communication costs. While this is probably reasonable at a relatively small number of parallel processes (i.e., that can be on a single multi-processor node of a cluster), at larger scales it may become material and worth trying to minimize.
- Adapting our methods to approximate tensor network contraction.
- Finding analogous methods for optimizing tensor-network ansatzes. For example, it is known that optimizing (bounded-bond dimension) tree tensor networks is easy. Can this be generalized in a parameterizable way as we did for tensor network contraction?

References

1 Scott Aaronson and Lijie Chen. Complexity-theoretic foundations of quantum supremacy experiments. *arXiv preprint*, 2016. arXiv:1612.05903.

2 The Parameterized Algorithms and Computational Experiments Challenge. Track A: Treewidth, December 2016. URL: https://pacechallenge.wordpress.com/pace-2017/track-a-treewidth/.

3 Itai Arad and Zeph Landau. Quantum computation and the evaluation of tensor networks. *SIAM Journal on Computing*, 39(7):3089–3121, 2010.

4 Stefan Arnborg and Andrzej Proskurowski. Linear time algorithms for NP-hard problems restricted to partial k-trees. *Discrete applied mathematics*, 23(1):11–24, 1989.

5 Jacob Biamonte and Ville Bergholm. Tensor Networks in a Nutshell. *arXiv e-prints*, page arXiv:1708.00006, July 2017. arXiv:1708.00006.

6 Jacob D. Biamonte, Jason Morton, and Jacob Turner. Tensor Network Contractions for #SAT. *Journal of Statistical Physics*, 160(5):1389–1404, September 2015. doi:10.1007/s10955-015-1276-z.

7 Dan Bienstock. On embedding graphs in trees. *Journal of Combinatorial Theory, Series B*, 49(1):103–136, 1990.

8 Hans L. Bodlaender. A Linear Time Algorithm for Finding Tree-decompositions of Small Treewidth. In *Proceedings of the Twenty-fifth Annual ACM Symposium on Theory of Computing*, STOC '93, pages 226–234, New York, NY, USA, 1993. ACM. doi:10.1145/167088.167161.

9 Hans L Bodlaender, Michael R Fellows, and Dimitrios M Thilikos. Derivation of algorithms for cutwidth and related graph layout parameters. *Journal of Computer and System Sciences*, 75(4):231–244, 2009.

10 Hans L. Bodlaender, John R. Gilbert, Hjálmtýr Hafsteinsson, and Ton Kloks. Approximating treewidth, pathwidth, and minimum elimination tree height. In Gunther Schmidt and Rudolf Berghammer, editors, *Graph-Theoretic Concepts in Computer Science*, pages 1–12, Berlin, Heidelberg, 1992. Springer Berlin Heidelberg.

11 Hans L Bodlaender, John R Gilbert, Hjálmtyr Hafsteinsson, and Ton Kloks. Approximating treewidth, pathwidth, frontsize, and shortest elimination tree. *J. Algorithms*, 18(2):238–255, 1995.

12 Hans L. Bodlaender and Dimitrios M. Thilikos. Constructive linear time algorithms for branchwidth. In Pierpaolo Degano, Roberto Gorrieri, and Alberto Marchetti-Spaccamela, editors, *Automata, Languages and Programming*, pages 627–637, Berlin, Heidelberg, 1997. Springer Berlin Heidelberg.

13 Sergio Boixo, Sergei V Isakov, Vadim N Smelyanskiy, and Hartmut Neven. Simulation of low-depth quantum circuits as complex undirected graphical models. *arXiv preprint*, 2017. arXiv:1712.05384.

14 Jianxin Chen, Fang Zhang, Mingcheng Chen, Cupjin Huang, Michael Newman, and Yaoyun Shi. Classical simulation of intermediate-size quantum circuits. *arXiv preprint*, 2018. arXiv:1805.01450.

15 William Cook and Paul Seymour. Tour Merging via Branch-Decomposition. *INFORMS Journal on Computing*, 15(3):233–248, 2003. doi:10.1287/ijoc.15.3.233.16078.

16 Marek Cygan, Fedor V. Fomin, Łukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michał Pilipczuk, and Saket Saurabh. *Lower Bounds Based on the Exponential-Time Hypothesis*, pages 467–521. Springer International Publishing, Cham, 2015. doi:10.1007/978-3-319-21275-3_14.

17 Reinhard Diestel. *Graph theory*. Springer Publishing Company, Incorporated, 2018.

18 Eugene Dumitrescu. Tree tensor network approach to simulating Shor's algorithm. *Phys. Rev. A*, 96:062322, December 2017. doi:10.1103/PhysRevA.96.062322.

19 Eugene F. Dumitrescu, Allison L. Fisher, Timothy D. Goodrich, Travis S. Humble, Blair D. Sullivan, and Andrew L. Wright. Benchmarking treewidth as a practical component of tensor

network simulations. *PLOS ONE*, 13(12):1–19, December 2018. doi:10.1371/journal.pone.0207827.

20 E. Schuyler Fried, Nicolas P. D. Sawaya, Yudong Cao, Ian D. Kivlichan, Jhonathan Romero, and Alán Aspuru-Guzik. qTorch: The quantum tensor contraction handler. *PLOS ONE*, 13(12):1–20, December 2018. doi:10.1371/journal.pone.0208510.

21 Daniel J. Harvey and David R. Wood. The treewidth of line graphs. *Journal of Combinatorial Theory, Series B*, 132:157–179, 2018. doi:10.1016/j.jctb.2018.03.007.

22 I. Markov and Y. Shi. Simulating Quantum Computation by Contracting Tensor Networks. *SIAM Journal on Computing*, 38(3):963–981, 2008. doi:10.1137/050644756.

23 M.I Ostrovskii. Minimal congestion trees. *Discrete Mathematics*, 285(1):219–226, 2004. doi:10.1016/j.disc.2004.02.009.

24 Edwin Pednault, John A. Gunnels, Giacomo Nannicini, Lior Horesh, Thomas Magerlein, Edgar Solomonik, Erik W. Draeger, Eric T. Holland, and Robert Wisnieff. Breaking the 49-Qubit Barrier in the Simulation of Quantum Circuits. *arXiv e-prints*, page arXiv:1710.05867, October 2017. arXiv:1710.05867.

25 Neil Robertson and P.D Seymour. Graph minors. X. Obstructions to tree-decomposition. *Journal of Combinatorial Theory, Series B*, 52(2):153–190, 1991. doi:10.1016/0095-8956(91)90061-N.

26 P. D. Seymour and R. Thomas. Call routing and the ratcatcher. *Combinatorica*, 14(2):217–241, June 1994. doi:10.1007/BF01215352.

27 Benjamin Villalonga, Sergio Boixo, Bron Nelson, Christopher Henze, Eleanor Rieffel, Rupak Biswas, and Salvatore Mandrà. A flexible high-performance simulator for the verification and benchmarking of quantum circuits implemented on real hardware. *arXiv e-prints*, page arXiv:1811.09599, November 2018. arXiv:1811.09599.

A Branchwidth and edge congestion

Proof of Theorem 12. First, we show how to compute a branch decomposition of $L(G)$ with width k given a tree embedding of G with congestion k, implying $\mathrm{bw}(L(G)) \leq \mathrm{ec}(G)$. Suppose we have a tree embedding (T, b) of G with edge congestion k. Let T' be a copy of T and $b' : E(L(G)) \to V(T)$ be a mapping from edges of the line graph to vertices of T. In particular, for an edge $e = \{\{u, v\}, \{v, w\}\}$ of $L(G)$ set $b' : \{\{u, v\}, \{v, w\}\} \mapsto b(v)$, i.e., b' maps adjacent pairs of edges of G to the same leaf mapped to from their common vertex by b. Interpreted as a branch decomposition of $L(G)$, (T', b') has width k, except that b' is not injective. We will now introduce a series of modifications to (T', b') that will turn it into a proper branch decomposition with width k. Note that $b'(e) = b'(f)$ if and only if e and f correspond to the same vertex of G. For each vertex v of G, we will replace the corresponding leaf of T with a subtree whose leaves are one-to-one with the edges of $E(L(G))$ corresponding to the vertex v. Consider a particular vertex v. Let l_0 be the corresponding leaf of T and t_0 its neighbor. Let $(e_1, e_2, \ldots, e_{\deg_G(v)})$ be an arbitrary ordering of the edges adjacent to v in G. First, we replace the leaf l_0 with a subcubic caterpillar graph with internal vertices $(t_1, t_2, \ldots, t_{\deg_G(v)-2})$ and leaves $(l_1, l_2, \ldots, l_{\deg_G(v)-1})$ such that t_i is adjacent to t_{i-1} and l_i for $i \in [\deg_G(v) - 2]$ and $t_{\deg_G(v)-2}$ is adjacent to $l_{\deg_G(v)-1}$. Then we set $b' : \{e_i, e_j\} \mapsto l_{\min(i,j)}$.

At this point $\left|b'^{-1}(l_i)\right| = |\{\{e_i, e_j\} | j > i\}| = \deg_G(v) - i$. For each l_i we do the following. Relabel its neighbor t_i as $t_{i,0}$. Replace l_i with another subcubic caterpillar graph with internal vertices $\left(t_{i,1}, t_{i,2}, t_{i,\deg_G(v)-i-2}\right)$ and leaves $\left(l_{i,1}, l_{i,2}, l_{i,\deg_G(v)-i-1}\right)$ such that $t_{i,j}$ is adjacent to $l_{i,j}$ and $t_{i,j-1}$ for $j \in [\deg_G(v) - 2]$ and $t_{i,\deg_G(v)-2}$ is adjacent to $l_{i,\deg_G(v)-1}$.

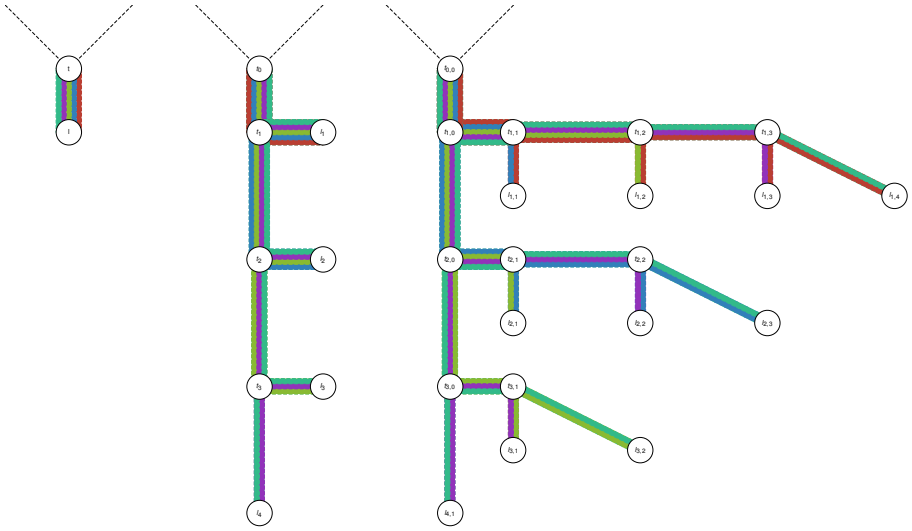


Figure 7 From a tree embedding of G to a branch decomposition of $L(G)$. Left: a leaf l_0 and neighboring vertex t_0 of a tree embedding. Middle: Replacement of the leaf l_0 with a caterpillar subtree. Right: Replacement of each leaf with a caterpillar subtree.

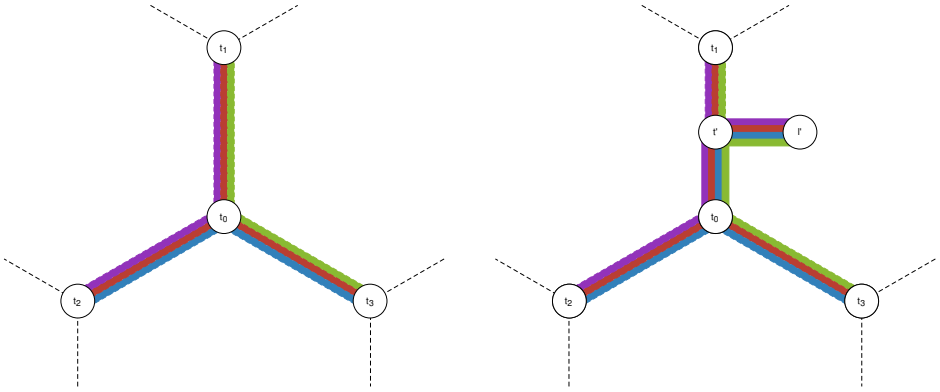


Figure 8 From a branch decomposition of $L(G)$ to a tree embedding of G. Left: Part of a branch decomposition. Right: Modified part to form a tree embedding.

Then set

$$b' : \{e_i, e_j\} \mapsto \begin{cases} l_{i,j-i}, & i < j, \\ l_{j,i-j}, & i > j. \end{cases} \tag{2}$$

At this point, (T', b') is a proper branch decomposition of the line graph $L(G)$. What is its width? Let S'_e be the subtree connecting $b'(\{e, f\})$ for all neighbors f of e in $L(G)$. In the part of T' that we didn't change, this coincides with S_e of the tree embedding (T, b). The number of subtrees including the edge $\{t_0, t_{1,0}\}$ of T' is the same as that including the edge $\{t_0, l_0\}$ of T, which is at most the edge congestion of (T, b). In particular, it is exactly $\deg_G(v)$. These are the only subtrees that contain any part of the new parts of the tree T' that we created. The contstruction is shown for a degree 5 vertex in Figure 7.

Now, we show how to compute a tree embedding with congestion k from a width-k branch decomposition the line graph, implying $\mathrm{ec}(G) \leq \mathrm{bw}(G)$. Suppose we have a width-k branch decomposition (T, b) of $L(G)$. Let T' be a tree and b' a function from $V(G)$ to $V(T')$. Initially we set $(T', b') = (T, b)$ and iteratively modify it into a tree embedding. For each vertex $v \in V(G)$, the neighboring edges $E_v \subset E(G) = V(L(G))$ form a clique of size $\deg_G(v)$. Therefore, there must be some vertex t_0 of T such that S_e contains t_0 for all $e \in E_v$. Let t_1, t_2, t_3 be the three neighbors of t_0 and partition E_v into four (potentially empty) parts: E_0 contains those edges e such that S_e contains all of t_0, t_1, t_2, t_3 and E_i contains those edges e such that S_e does not contain t_i, for $i = 1, 2, 3$. Without loss of generality, assume $w(E_1) \leq w(E_2) \leq w(E_3)$. Note that $\deg(v) = \sum_{i=0}^{4} w(E_i) \geq \sum_{i=1}^{4} w(E_i) \geq 3w(E_1)$. Now, subdivide the edge between t_0 and t_1, introducing a new vertex t', and add a new leaf l' adjacent thereto. For all $e \in E_v$, set $b'(e) = l'$; this leaf will correspond to vertex v in the tree embedding. Note that the congestion of the edge between l' and t' is $\deg(v)$, and that the congestion of the edge between t' and t_0 is $w(E_1) \leq \deg(v)/3$ more than the congestion of the edge $\{t_0, t_1\}$ that it replaced. If we do this for every vertex, we get a tree embedding whose congestion is at most $\deg(G)/3$ more than the width of the branch decomposition we started with. This is illustrated in Figure 8. ◀

It cannot be the case that for every graph G, $\mathrm{bw}(L(G)) = \mathrm{ec}(G)$. Consider, for example, the star graph S_k. Its edge congestion is at least its maximum degree k, but its line graph is the complete graph, whose branchwidth is $\lceil \frac{2k}{3} \rceil$.

Consider an alternative, what we'll call the *line hypergraph*, denoted $L^*(G)$, with a vertex for each edge of $E(G)$ and a *hyperedge* for each vertex of $V(G)$ (rather than a clique as in the usual line graph). Then it is trivially true that $\mathrm{bw}(L^*(G)) = \mathrm{ec}(G)$.