

17th Conference on the Theory of Quantum Computation, Communication and Cryptography

TQC 2022, July 11–15, 2022, Urbana Champaign, Illinois, USA

Edited by

François Le Gall
Tomoyuki Morimae



Editors

François Le Gall

Graduate School of Mathematics, Nagoya University, Nagoya, Japan
legall@math.nagoya-u.ac.jp

Tomoyuki Morimae

Yukawa Institute for Theoretical Physics, Kyoto University, Kyoto, Japan
tomoyuki.morimae@yukawa.kyoto-u.ac.jp

ACM Classification 2012

Theory of computation → Quantum computation theory; Theory of computation → Quantum complexity theory; Theory of computation → Quantum information theory; Theory of computation → Quantum communication complexity; Hardware → Quantum communication and cryptography; Hardware → Quantum error correction and fault tolerance

ISBN 978-3-95977-237-2

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/978-3-95977-237-2>.

Publication date

July, 2022

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>.

License

This work is licensed under a Creative Commons Attribution 4.0 International license (CC-BY 4.0): <https://creativecommons.org/licenses/by/4.0/legalcode>.



In brief, this license authorizes each and everybody to share (to copy, distribute and transmit) the work under the following conditions, without impairing or restricting the authors' moral rights:

- Attribution: The work must be attributed to its authors.

The copyright is retained by the corresponding authors.

Digital Object Identifier: 10.4230/LIPIcs.TQC.2022.0

ISBN 978-3-95977-237-2

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.

Editorial Board

- Luca Aceto (*Chair*, Reykjavik University, IS and Gran Sasso Science Institute, IT)
- Christel Baier (TU Dresden, DE)
- Mikolaj Bojanczyk (University of Warsaw, PL)
- Roberto Di Cosmo (Inria and Université de Paris, FR)
- Faith Ellen (University of Toronto, CA)
- Javier Esparza (TU München, DE)
- Daniel Král' (Masaryk University - Brno, CZ)
- Meena Mahajan (Institute of Mathematical Sciences, Chennai, IN)
- Anca Muscholl (University of Bordeaux, FR)
- Chih-Hao Luke Ong (University of Oxford, GB)
- Phillip Rogaway (University of California, Davis, US)
- Eva Rotenberg (Technical University of Denmark, Lyngby, DK)
- Raimund Seidel (Universität des Saarlandes, Saarbrücken, DE and Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Wadern, DE)

ISSN 1868-8969

<https://www.dagstuhl.de/lipics>

■ Contents

| | |
|--|-------|
| Preface | |
| <i>François Le Gall and Tomoyuki Morimae</i> | 0:vii |
| Conference Organization | |
| | 0:ix |
| List of Authors | |
| | 0:xi |

Papers

| | |
|--|------------|
| Quantum Algorithms for Learning a Hidden Graph | |
| <i>Ashley Montanaro and Changpeng Shao</i> | 1:1–1:22 |
| Quantum Algorithm for Stochastic Optimal Stopping Problems with Applications in Finance | |
| <i>João F. Doriguello, Alessandro Luongo, Jinge Bao, Patrick Reberntrost, and Miklos Santha</i> | 2:1–2:24 |
| The Parametrized Complexity of Quantum Verification | |
| <i>Srinivasan Arunachalam, Sergey Bravyi, Chinmay Nirkhe, and Bryan O’Gorman</i> | 3:1–3:18 |
| Averaged Circuit Eigenvalue Sampling | |
| <i>Steven T. Flammia</i> | 4:1–4:10 |
| Classical Simulation of Quantum Circuits with Partial and Graphical Stabiliser Decompositions | |
| <i>Aleks Kissinger, John van de Wetering, and Renaud Vilmart</i> | 5:1–5:13 |
| On Converses to the Polynomial Method | |
| <i>Jop Briët and Francisco Escudero Gutiérrez</i> | 6:1–6:10 |
| The Quantum Approximate Optimization Algorithm at High Depth for MaxCut on Large-Girth Regular Graphs and the Sherrington-Kirkpatrick Model | |
| <i>Joao Basso, Edward Farhi, Kunal Marwaha, Benjamin Villalonga, and Leo Zhou</i> | 7:1–7:21 |
| A Constant Lower Bound for Any Quantum Protocol for Secure Function Evaluation | |
| <i>Sarah A. Osborn and Jamie Sikora</i> | 8:1–8:14 |
| Approximating Output Probabilities of Shallow Quantum Circuits Which Are Geometrically-Local in Any Fixed Dimension | |
| <i>Suchetan Dontha, Shi Jie Samuel Tan, Stephen Smith, Sangheon Choi, and Matthew Coudron</i> | 9:1–9:17 |
| Memory Compression with Quantum Random-Access Gates | |
| <i>Harry Buhrman, Bruno Loff, Subhasree Patro, and Florian Speelman</i> | 10:1–10:19 |

17th Conference on the Theory of Quantum Computation, Communication and Cryptography (TQC 2022).
Editors: François Le Gall and Tomoyuki Morimae



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

0:vi **Contents**

Quantum Speedups for Treewidth
Vladislavs Kļevickis, Krišjānis Prūsis, and Jevgēnijs Vihrovs 11:1–11:18

Qutrit Metaplectic Gates Are a Subset of Clifford+T
Andrew N. Glauddell, Neil J. Ross, John van de Wetering, and Lia Yeh 12:1–12:15

■ Preface

The 17th Conference on the Theory of Quantum Computation, Communication and Cryptography was hosted by the University of Illinois at Urbana-Champaign, and held from July 11 to July 15, 2022.

Quantum computation, quantum communication, and quantum cryptography are subfields of quantum information processing, an interdisciplinary field of information science and quantum mechanics. The TQC conference series focuses on theoretical aspects of these subfields. The objective of the conference is to bring together researchers so that they can interact with each other and share problems and recent discoveries.

A list of the previous editions of TQC follows:

- TQC 2021, University of Latvia, Latvia (virtual conference)
- TQC 2020, University of Latvia, Latvia (virtual conference)
- TQC 2019, University of Maryland, USA
- TQC 2018, University of Technology Sydney, Australia
- TQC 2017, Université Pierre et Marie Curie, France
- TQC 2016, Freie Universität Berlin, Germany
- TQC 2015, Université libre de Bruxelles, Brussels, Belgium
- TQC 2014, National University of Singapore, Singapore
- TQC 2013, University of Guelph, Canada
- TQC 2012, University of Tokyo, Japan
- TQC 2011, Universidad Complutense de Madrid, Spain
- TQC 2010, University of Leeds, UK
- TQC 2009, Institute for Quantum Computing, University of Waterloo, Canada
- TQC 2008, University of Tokyo, Japan
- TQC 2007, Nara Institute of Science and Technology, Nara, Japan
- TQC 2006, NTT R&D Center, Atsugi, Kanagawa, Japan

We wish to thank the members of the Program Committee and all subreviewers for their precious help. Our warm thanks also go to the members of the Local Organizing Committee, for their considerable efforts in organizing the conference. We would like to thank the members of the Steering Committee for giving us the opportunity to work for TQC. And, of course, all contributors and participants!

April 2022

François Le Gall and Tomoyuki Morimae



■ Conference organization

Local Organizing Committee

- Eric Chitambar (University of Illinois at Urbana-Champaign – chair)
- Emily Edwards (University of Illinois at Urbana-Champaign)
- Kim Gudeman (University of Illinois at Urbana-Champaign)
- Felix Leditzky (University of Illinois at Urbana-Champaign)
- Hannah Stites (University of Illinois at Urbana-Champaign)

Program Committee

- Dominic Berry (Macquarie University)
- Mario Berta (AWS Center for Quantum Computing & Imperial College London)
- Dan Browne (University College London)
- Francesco Buscemi (Nagoya University)
- Marco Cerezo (Los Alamos National Laboratory)
- Kai-Min Chung (Academia Sinica)
- Wim van Dam (University of California, Santa Barbara)
- Jens Eisert (Freie Universität Berlin)
- David Elkouss (QuTech)
- Bill Fefferman (The University of Chicago)
- Zoe Holmes (Los Alamos National Laboratory)
- Isaac Kim (Stanford)
- Robert Koenig (Technische Universität München)
- Richard Kueng (Johannes Kepler University Linz)
- Sophie Laplante (Université Paris Cité)
- François Le Gall (Nagoya University – chair)
- Felix Leditzky (University of Illinois at Urbana-Champaign)
- Tongyang Li (Peking University)
- Qipeng Liu (Simons Institute for the Theory of Computing)
- Frédéric Magniez (CNRS)
- Christian Majenz (Technical University of Denmark)
- Carl Miller (NIST and University of Maryland)
- Tomoyuki Morimae (Kyoto University – co-chair)
- Ashwin Nayak (University of Waterloo)
- Nelly Ng (Nanyang Technological University)
- Harumichi Nishimura (Nagoya University)
- Michał Oszmaniec (CTP PAS)
- Anna Pappa (Technische Universität Berlin)
- Simon Perdrix (Inria LORIA)
- Stefano Pironio (Université libre de Bruxelles)
- Patrick Reberntrost (CQT)
- Ben Reichardt (University of Southern California)
- Yuval Rishu Sanders (University of Technology Sydney)
- Norbert Schuch (University of Vienna)
- Yuan Su (Google Quantum AI)

17th Conference on the Theory of Quantum Computation, Communication and Cryptography (TQC 2022).
Editors: François Le Gall and Tomoyuki Morimae



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

0:x Conference organization

- Aarthi Sundaram (Microsoft)
- Penghui Yao (Nanjing University)


Steering Committee


- Gorjan Alagic (Maryland)
- Andris Ambainis (Latvia)
- Eric Chitambar (University of Illinois at Urbana-Champaign)
- Steve Flammia (Sydney)
- Stacey Jeffery (QuSoft, CWI)
- Min-Hsiu Hsieh (Hon Hai)
- Laura Mančinska (Copenhagen)
- Marco Tomamichel (National University of Singapore – chair)

■ List of Authors

Srinivasan Arunachalam (3)
IBM Quantum, Thomas J Watson Research
Center, Yorktown Heights, NY, USA

Jinge Bao (2)
Centre for Quantum Technologies, National
University of Singapore, Singapore

Joao Basso  (7)
Google Quantum AI, Venice, CA, USA

Sergey Bravyi  (3)
IBM Quantum, Thomas J Watson Research
Center, Yorktown Heights, NY, USA


Jop Briët (6)
CWI & QuSoft, Amsterdam, The Netherlands

Harry Buhrman (10)
QuSoft, CWI Amsterdam, The Netherlands;
University of Amsterdam, The Netherlands

Sangheon Choi (9)
Department of Computer Science, Rose-Hulman
Institute of Technology, Terre Haute, IN, USA


Matthew Coudron (9)
Department of Computer Science,
University of Maryland, College Park, MD, USA


Suchetan Dontha (9)
Department of Computer Science,
University of Maryland, College Park, MD, USA


João F. Doriguello  (2)
Centre for Quantum Technologies, National
University of Singapore, Singapore

Francisco Escudero Gutiérrez (6)
CWI & QuSoft, Amsterdam, The Netherlands

Edward Farhi (7)
Google Quantum AI, Venice, CA, USA;
Center for Theoretical Physics, Massachusetts
Institute of Technology, Cambridge, MA, USA

Steven T. Flammia  (4)
AWS Center for Quantum Computing,
Pasadena, CA, USA;
California Institute of Technology,
Pasadena, CA, USA


Andrew N. Glaudell  (12)
Booz Allen Hamilton, Atlanta, GA, USA;
Department of Mathematics, George Mason
University, Fairfax, VA, USA

Aleks Kissinger  (5)
University of Oxford, UK


Vladislavs Kļevickis (11)
Centre for Quantum Computer Science, Faculty
of Computing, University of Latvia, Riga, Latvia

Bruno Loff (10)
University of Porto, Portugal;
INESC-Tec, Porto, Portugal

Alessandro Luongo (2)
Centre for Quantum Technologies, National
University of Singapore, Singapore

Kunal Marwaha  (7)
Department of Computer Science,
University of Chicago, IL, USA

Ashley Montanaro (1)
School of Mathematics,
University of Bristol, UK;
Phasecraft Ltd., Bristol, UK

Chinmay Nirkhe  (3)
IBM Quantum, Thomas J Watson Research
Center, Yorktown Heights, NY, USA;
Electrical Engineering and Computer Sciences,
University of California, Berkeley, CA, USA;
Challenge Institute for Quantum Computation,
University of California, Berkeley, CA, USA


Bryan O’Gorman  (3)
IBM Quantum, Thomas J Watson Research
Center, Yorktown Heights, NY, USA

Sarah A. Osborn (8)
Virginia Polytechnic Institute and State
University, Blacksburg, VA, USA

Subhasree Patro (10)
QuSoft, CWI Amsterdam, The Netherlands;
University of Amsterdam, The Netherlands

Krišjānis Prūsis (11)
Centre for Quantum Computer Science, Faculty
of Computing, University of Latvia, Riga, Latvia

Patrick Rebentrost (2)
Centre for Quantum Technologies, National
University of Singapore, Singapore

Neil J. Ross  (12)
Department of Mathematics and Statistics,
Dalhousie University, Halifax, Canada

17th Conference on the Theory of Quantum Computation, Communication and Cryptography (TQC 2022).
Editors: François Le Gall and Tomoyuki Morimae



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

- Miklos Santha (2)
Centre for Quantum Technologies, National
University of Singapore, Singapore
- Changpeng Shao (1)
School of Mathematics,
University of Bristol, UK
- Jamie Sikora (8)
Virginia Polytechnic Institute and State
University, Blacksburg, VA, USA
- Stephen Smith (9)
Department of Mathematics, University of South
Carolina, Columbia, SC, USA
- Florian Speelman (10)
QuSoft, CWI Amsterdam, The Netherlands;
University of Amsterdam, The Netherlands
- Shi Jie Samuel Tan (9)
Department of Computer Science,
Haverford College, PA, USA
- John van de Wetering  (5, 12)
Radboud University Nijmegen, The Netherlands;
University of Oxford, United Kingdom
- Jevgēnijs Vihrovs  (11)
Centre for Quantum Computer Science, Faculty
of Computing, University of Latvia, Riga, Latvia
- Benjamin Villalonga  (7)
Google Quantum AI, Venice, CA
- Renaud Vilmart  (5)
Université Paris-Saclay, CNRS, ENS
Paris-Saclay, Inria, Laboratoire Méthodes
Formelles, 91190, Gif-sur-Yvette, France
- Lia Yeh  (12)
Department of Computer Science,
University of Oxford, UK
- Leo Zhou  (7)
Walter Burke Institute for Theoretical Physics,
California Institute of Technology, Pasadena,
CA, USA

Quantum Algorithms for Learning a Hidden Graph

Ashley Montanaro ✉

School of Mathematics, University of Bristol, UK
Phasecraft Ltd., Bristol, UK

Changpeng Shao ✉

School of Mathematics, University of Bristol, UK

Abstract

We study the problem of learning an unknown graph provided via an oracle using a quantum algorithm. We consider three query models. In the first model (“OR queries”), the oracle returns whether a given subset of the vertices contains any edges. In the second (“parity queries”), the oracle returns the parity of the number of edges in a subset. In the third model, we are given copies of the graph state corresponding to the graph.

We give quantum algorithms that achieve speedups over the best possible classical algorithms in the OR and parity query models, for some families of graphs, and give quantum algorithms in the graph state model whose complexity is similar to the parity query model. For some parameter regimes, the speedups can be exponential in the parity query model. On the other hand, without any promise on the graph, no speedup is possible in the OR query model.

A main technique we use is the quantum algorithm for solving the combinatorial group testing problem, for which a query-efficient quantum algorithm was given by Belovs. Here we additionally give a time-efficient quantum algorithm for this problem, based on the algorithm of Ambainis et al. for a “gapped” version of the group testing problem.

2012 ACM Subject Classification Theory of computation → Quantum query complexity

Keywords and phrases Quantum algorithms, query complexity, graphs, combinatorial group testing

Digital Object Identifier 10.4230/LIPIcs.TQC.2022.1

Funding This paper was supported by the QuantERA ERA-NET Cofund in Quantum Technologies implemented within the European Union’s Horizon 2020 Programme (QuantAlgo project) and EPSRC grants EP/R043957/1 and EP/T001062/1. This project has received funding from the European Research Council (ERC) under the European Union’s Horizon 2020 research and innovation programme (grant agreement No. 817581).

Acknowledgements We would like to thank João Doriguello and Ryan Mann for helpful discussions on the topic of this work.

1 Introduction

Quantum computers are known to be able to compute certain functions more quickly than their classical counterparts, in terms of the number of queries to the input that are required. In some cases, quantum algorithms can also learn unknown objects using fewer queries than their classical counterparts. For example, if we are given query access to an unknown boolean function on n -bits which is promised to be a dot product between x and a secret string s modulo 2, then the Bernstein-Vazirani algorithm learns this function with 1 query [17], while the best possible classical algorithm uses n queries. If the function is promised to be an OR function of k unknown variables, then Belovs’ algorithm for combinatorial group testing [15] learns this function with $\Theta(\sqrt{k})$ queries, while the best possible classical algorithm needs $\Theta(k \log(n/k))$ queries. These speedups are not far from the largest quantum speedups that can be achieved. For any class \mathcal{C} of Boolean functions over $\{0, 1\}^n$, let D and Q be such that an unknown function from \mathcal{C} can be identified using D classical membership queries or from Q quantum membership queries. Then $D = O(nQ^3)$ [41].



© Ashley Montanaro and Changpeng Shao;
licensed under Creative Commons License CC-BY 4.0

17th Conference on the Theory of Quantum Computation, Communication and Cryptography (TQC 2022).

Editors: François Le Gall and Tomoyuki Morimae; Article No. 1; pp. 1:1–1:22

Leibniz International Proceedings in Informatics



LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

■ **Table 1** Query complexities for learning various classes of graphs: m is the number of edges, n is the number of all vertices. The symbols \vee (OR), \oplus (parity), and $|G\rangle$ (graph state) denote the type of query considered. Q and C denote quantum and classical queries.

| | Q, \vee | Q, \oplus | C, \vee, \oplus | Q, $ G\rangle$ |
|--------------------|---|-------------------------|--------------------------------|---------------------------|
| All graphs | $\Theta(n^2)$ | $\Theta(n)$ | $\Theta(n^2)$ | $\Theta(n)$ |
| m edges | $O(m \log(\sqrt{m} \log n) + \sqrt{m} \log n)$ | $O(\sqrt{m \log m})$ | $\Omega(m \log \frac{n^2}{m})$ | $O(m \log \frac{n^2}{m})$ |
| Degree d | $O(d^4 m^{3/4} \sqrt{\log n} (\log m) + \sqrt{m} \log n)$ | $O(d \log \frac{m}{d})$ | $\Omega(nd \log \frac{n}{d})$ | $O(d \log \frac{m}{d})$ |
| Matching | $O(m^{3/4} \sqrt{(\log n)(\log m)} + \sqrt{m} \log n)$ | $O(\log m)$ | $\Omega(m \log \frac{n}{m})$ | $O(\log m)$ |
| Cycle | $O(m^{3/4} \sqrt{(\log n)(\log m)} + \sqrt{m} \log n)$ | $O(\log m)$ | $\Omega(m \log \frac{n}{m})$ | $O(\log m)$ |
| Star | $\Theta(\sqrt{m})$ | $O(1)$ | $\Omega(m \log \frac{n}{m})$ | $O(1)$ |
| k -vertex clique | $\Theta(\sqrt{k})$ | $O(1)$ | $\Omega(k \log \frac{n}{k})$ | $O(1)$ |

Here we focus on the problem of learning an unknown graph using quantum queries, in a variety of settings. Many quantum speedups (both polynomial, e.g. [27], and exponential, e.g. [16]) are known for problems involving graphs. However, the only quantum speedup we are aware of for *learning* graphs is recent work on learning graphs using cut queries [34].

We consider several different notions of queries to an unknown graph – OR queries, parity queries, and graph states, all defined below – and aim to minimize the number of queries required to identify the graph. The first two of these query models are closely related to models that have been extensively studied in the classical literature on exact learning, e.g. [9, 25, 31], in particular because of their applications to computational biology. In some cases we find polynomial speedups over the best possible classical complexity, while in other cases (such as learning bounded-degree graphs in the parity query model) the speedups can even be exponential. A summary of our results is as follows; also see Table 1. Throughout, we use n to denote the number of vertices and m to denote the number of edges of a graph.

1. **(OR queries)** First, we consider the problem of identifying an unknown graph, given access to queries to subsets of the vertices, which return whether the corresponding induced subgraph has any edges within that subset. That is, given a graph $G = (V, E)$, a query takes a subset $S \subseteq V$ and returns whether $E \cap (S \times S)$ is empty. This model has been extensively studied classically and we will briefly survey these results below. Our main results in this model are:
 - A quantum algorithm to learn an unknown graph with m edges using $O(m \log(\sqrt{m} \log n) + \sqrt{m} \log n)$ OR queries, as compared with the classical lower bound of $\Omega(m \log(n^2/m))$. For some relationships between m and n (e.g. $m = \Theta(\log n)$) this gives a modest quantum-classical separation.
 - The lower bound that any quantum algorithm that identifies an arbitrary unknown graph in this model must make $\Omega(n^2)$ OR queries, so the above algorithm’s complexity cannot be improved by more than log factors.
 - Learning graphs with special structure, such as Hamiltonian cycles, matchings, stars and cliques, has specific applications in molecular biology [5, 29, 30]. We give quantum speedups for learning these graphs in this model. The graphs and quantum speedups can be roughly summarized as follows. Hamiltonian cycles and matchings: $k^{3/4}$ vs. k ; stars and cliques: \sqrt{k} vs. k . Here k is the number of non-isolated vertices.
2. **(Parity queries)** Next, we consider the same problem, but where the oracle returns the *parity* of $|E \cap (S \times S)|$, for arbitrary subsets S . Although this may seem a more unusual setting, this oracle can be obtained from the perhaps more natural oracle, known

as additive oracle, that returns the size of $E \cap (S \times S)$, which has also been studied classically [18, 25, 31, 40]. We will see that larger quantum speedups are available in this model. Here, we show that:

- There is a quantum algorithm which learns an unknown graph with degree d making $O(d \log m)$ parity queries, as compared with the classical lower bound of $\Omega(nd \log(n/d))$ queries.
- There is a quantum algorithm which learns an unknown graph with m edges making $O(\sqrt{m \log m})$ parity queries, as compared with the classical lower bound of $\Omega(m \log(n^2/m))$.
- Stars and cliques can be learned with $O(1)$ parity queries.

Our results show that, for some families of graphs, parity queries can be exponentially more efficient than OR queries for quantum algorithms. The results we obtain are based on very similar ideas to a recent work by Lee, Santha and Zhang [34], which considered a related “cut query” model (see below).

3. (Graph states) We also study a quantum version of the problem of learning an unknown graph: the problem of learning an unknown graph state [32]. Graph states are a family of quantum states that have many important applications, in particular to measurement-based quantum computing. Any graph G has a corresponding graph state $|G\rangle$, and it is a natural question to ask how many copies of $|G\rangle$ are required to identify G . It was already known that $\Theta(n)$ copies are necessary and sufficient if G is an arbitrary graph with n vertices [1, 36, 42]. However, we show that one can do better given some additional information about G :

- If G has degree d , we can learn G using $O(d \log m)$ copies. If G is promised to be a subgraph of a known graph G' with bounded degree d , the quantum algorithm is also time-efficient (has runtime $\tilde{O}(d^3 n)$). This second algorithm could be particularly useful in the practically-relevant scenario where we aim to produce a desired graph state G' , but some edges of G' have failed to be generated, and we would like to determine which edges have failed.
- If G is known to be picked from a set of size L , we can learn G using $O(\log L)$ copies. For example, if G is known to have at most m edges, we can learn G using $O(m \log n)$ copies.

The results about learning graph states also underpin the results about learning graphs from parity queries, because it turns out that using a procedure known as Bell sampling [36] to learn a graph state is equivalent to learning a graph using parity queries – except with the restriction that these queries are only to uniformly random subsets of the vertices.

An important technique we use to learn a graph using the OR query is the quantum algorithm by Belovs [15] for combinatorial group testing (CGT, also known as “pooled testing”) [26, 39]. Belovs’ algorithms are produced by directly solving the semidefinite program for the general adversary bound, which is known to characterise quantum query complexity. This approach is beautiful but rather complex, and leads to algorithms which are not necessarily efficient in terms of time complexity. Here we give a quantum algorithm for CGT that makes $\tilde{O}(\sqrt{k})$ queries and runs in time $\tilde{O}(n\sqrt{k})$, based on the use of an algorithm of Ambainis et al. [6] for a “gapped” version of the group testing problem.

In Appendix B, we also give simple explicit quantum algorithms for learning an unknown subset on which the exact-half function or majority function acts, which match the complexity of previous algorithms by Belovs.

1.1 Summary of the techniques

The OR query model. In this model, we use a similar strategy to the classical algorithm given by Angluin and Chen [9]. The basic idea of [9] is binary search: We decompose the set of vertices V into halves V_1, V_2 , and suppose we already know the edges in V_1, V_2 . We then try to learn the edges between them. The edges in V_1, V_2 can be learned recursively, and the complexity is dominated by the learning of the edges between V_1, V_2 . This is an adaptive algorithm. On a quantum computer, we can use the quantum algorithm for CGT to accelerate the learning of the edges between V_1, V_2 . However, the classical inductive idea may not be applicable to the quantum case. A reason is that the underlying constant in the complexity of quantum algorithm for CGT is unknown for us, so we cannot bound the overall complexity easily. To overcome this problem, we first decompose V into a disjoint union of some subsets such that each subset contains no edges, then learn the edges between the subsets. This idea is inspired by the non-adaptive learning algorithm of [9].

The graph state model. We can apply Bell sampling to learn an unknown graph state [36]. Each Bell sample returns a uniformly random stabilizer of the graph state. Equivalently, if A is the adjacency matrix of the graph, then each Bell sample returns $As \pmod{2}$ for a random vector $s \in \{0, 1\}^n$. If we take k samples, then we obtain an $n \times k$ matrix B and the matrix AB . From B, AB we can determine A by choosing a suitable k .

The parity query model. Since the graph state can be generated by a parity query on a uniform superposition, any results for the graph state model also hold for the parity query model. Differently from the graph state model, with parity queries, we do have control of s . More precisely, for any $s \in \{0, 1\}^n$, there is a quantum algorithm that returns $As \pmod{2}$ using two parity queries. With this result, we can learn graphs of m edges more efficiently by considering the low and high-degree parts.

1.2 Prior work

Learning graphs with OR queries. Graph learning appears in many different contexts. In different applications, we apply different queries, and the OR query is important for problems in computational biology. This type of query is also known as independent set query [13] and edge-detection query [9]. Many classical algorithms were discovered to learn graphs using OR queries in the past decades. For special graphs, Beigel et al. [14] and Alon et al. [5] have given algorithms for learning an unknown matching using $O(n \log n)$ queries. Grebinski and Kucherov [29] gave an algorithm for learning a Hamiltonian cycle using $O(n \log n)$ queries. Alon and Asodi [4] gave bounds on nonadaptive deterministic algorithms for learning stars and cliques. Bouvel et al. [18] gave algorithms for learning an unknown star or clique using $O(n)$ queries. The constant factors in the algorithms for learning Hamiltonian cycles, matchings, stars and cliques were improved by Chang et al. [22].

In the general case, Angluin and Chen [9] gave a deterministic adaptive algorithm with complexity $O(m \log n)$ for learning a graph with m edges, encompassing all the above bounds (however, note that other restrictions can be considered, such as nonadaptivity, or restricted levels of adaptivity). The constant factor in this runtime was improved by Chang, Fu and Shih [23]. The complexity $O(m \log n)$ obtained in [9] assumes m is known in advance. When m is not known, the complexity of [9] is $O(m \log n + \sqrt{m} \log^2 n)$. This is recently improved to $O(m \log n + \sqrt{m}(\log n)(\log^{.k} \log n))$ in [2], where k can be any constant.

Graph states. In [42], Zhao, Pérez-Delgado and Fitzsimons studied the problem of representing basic operations of graphs by graph states with high efficiency and showed that no classical data structure can have similar performance. In this work, the authors gave an algorithm for learning an arbitrary graph state of n qubits using $O(n)$ copies. Graph states are a subclass of stabilizer states. Alternative algorithms for learning an arbitrary stabilizer state with $O(n)$ copies have been given by Aaronson and Gottesman [1] and Montanaro [36].

Learning graphs with parity queries. The parity query model is a special case of a model for graph queries which generalises the OR query model, and is known as additive queries [25,31] (also known as quantitative queries [18] and edge counting queries [40]). The additive query plays an important role for applications related to DNA sequencing. In this model, a query to a subset S returns the number of edges of G in S ; the parity query model is obtained if this answer is taken mod 2.

The additive query is known to be somewhat more powerful than the OR query for learning graphs. For instance, as shown in [18], a Hamiltonian cycle or a matching can be identified with $O(n)$ additive queries, while this requires at least $\Omega(n \log n)$ OR queries. Stars and cliques can be identified with $O(n/\log n)$ additive queries or with at least $\Omega(n)$ OR queries. Our results summarized in Table 1 also confirm that parity queries (and hence additive queries) are more powerful than OR queries in the quantum case. Some other results include the following. Graphs with maximum degree d can be learned with $O(dn)$ additive queries [31]. This is also true for learning bipartite graphs with maximum degree d non-adaptively [18]. Graphs with m edges can be learned with $O(m(\log n)/(\log m))$ additive queries [20,25].

Our results in the parity query model are closely related to a recent work by Lee, Santha and Zhang [34]. These authors showed that weighted graphs with maximum degree d can be learned using $O(d \log^2 n)$ quantum “cut queries”, and graphs with m edges can be learned using $O(\sqrt{m} \log^{3/2} n)$ quantum cut queries. A cut query takes as input a subset S of the vertices, and returns the number of edges of G with exactly one endpoint in S . Lee, Santha and Zhang also gave efficient quantum algorithms in this model for determining the number of connected components of G , and for outputting a spanning forest of G . It was shown in [34] that cut queries reduce to additive queries; however, there is no efficient reduction in the other direction. In [34, Corollary 27] stronger results than the cut-query results are given for additive queries: an $O(d \log(n/d))$ query algorithm for learning graphs with maximum degree d , and an $O(\sqrt{m \log n} + \log n)$ query algorithm for learning graphs with m edges. These algorithms are based on very similar ideas to the ones we state here (Theorems 14 and 17). Our algorithms as stated only require parity information (although the results of [34] could easily be rephrased in this way too); more importantly, the complexity of our results is somewhat better for graphs with very few edges, as a $\log n$ term is changed into a $\log m$ term. On the other hand, the algorithms of [34] are stated for the more general class of weighted graphs.

Combinatorial group testing (CGT). Classically, it is known that the number of queries required to solve CGT is $\Theta(k \log(n/k))$ [26]. In the quantum case, Ambainis and Montanaro [7] first studied this problem and proposed a quantum algorithm using $O(k)$ queries. They also showed a lower bound of $\Omega(\sqrt{k})$. Later in [15], based on the adversary bound method, Belovs proposed a quantum algorithm for CGT using $\Theta(\sqrt{k})$ queries.

1.3 Preliminaries

Let $f : \{0, 1\}^n \rightarrow \{0, 1\}$ be a boolean function with a quantum oracle to access it. That is, we are allowed to perform the map $|x\rangle|y\rangle \rightarrow |x\rangle|y \oplus f(x)\rangle$ for any $x \in \{0, 1\}^n, y \in \{0, 1\}$. Together with a simple phase flip unitary gate, we can also perform $|x\rangle \rightarrow (-1)^{f(x)}|x\rangle$. For any $x \in \{0, 1\}^n$, the Fourier coefficient of f at x is defined as $\widehat{f}(x) = \frac{1}{2^n} \sum_{s \in \{0, 1\}^n} (-1)^{f(s) + s \cdot x}$. We can equivalently associate each bit-string $x \in \{0, 1\}^n$ with a subset $S \subseteq [n]$. The Fourier sampling primitive is based on the following sequence of operations: First apply Hadamard gates $H^{\otimes n}$ to $|0\rangle^{\otimes n}$; then apply the oracle $|x\rangle \rightarrow (-1)^{f(x)}|x\rangle$; finally apply $H^{\otimes n}$ again. The resulting state is $\sum_{x \in \{0, 1\}^n} \widehat{f}(x)|x\rangle$. Measuring in the computational basis returns x with probability $\widehat{f}(x)^2$.

For an arbitrary graph $G = (V, E)$ on n vertices, and an arbitrary subset $S \subseteq V$, define the oracles $f_{G, \text{OR}}, f_{G, \text{Par}}$ by

- $f_{G, \text{OR}}(S) = 0$ if $|E \cap (S \times S)| = 0$, and $f_{G, \text{OR}}(S) = 1$ otherwise;
- $f_{G, \text{Par}}(S) = |\{E \cap (S \times S)\}| \bmod 2$.

We give quantum algorithms access to these oracles in the usual way described above.

A subroutine that will be used extensively throughout this paper is Belovs' efficient quantum algorithm for combinatorial group testing (CGT) [15]. In this problem, we are given oracle access to an n -bit string A with Hamming weight at most k . Usually, we assume that $k \ll n$. In one query, we can get the OR of an arbitrary subset of the bits of A . The goal is to determine A using the minimal number of queries. Belovs showed that this can be achieved using $O(\sqrt{k})$ quantum queries. For more details, refer to Appendix A.

► **Theorem 1** (Theorem 3.1 of [15]). *The quantum query complexity of the combinatorial group testing problem is $\Theta(\sqrt{k})$. The quantum algorithm succeeds with certainty.*

We sometimes use the notation $[X]$ for an expression which evaluates to 1 if X is true, and 0 if X is false.

2 Learning an unknown graph with OR queries

Let G be a graph with m edges and n vertices. Our goal is to identify all the edges in G using OR queries. We follow the same general strategy as Angluin and Chen [9] to achieve this by starting with special cases and progressively generalising. In particular, Lemmas 3 and 5 are direct quantum speedups of corresponding results (Lemmas 3.3 and 3.4) in [9]. The basic idea of the quantum learning algorithm is as follows: We first decompose the set of vertices into a disjoint union of several subsets. Each subset contains no edges. Then we learn the edges between these subsets. A sub-routine of this learning procedure is the quantum algorithm for solving solving combinatorial group testing (CGT), i.e., Theorem 1. It is the main ingredient to obtain quantum speedups.

Suppose A, B are two known, nonempty, independent (i.e., contain no edges) subsets of the set of vertices. The following lemma helps us efficiently identify the non-isolated vertices (those which have at least one edge incident to them).

► **Lemma 2.** *Assume that A and B are two known, disjoint, non-empty independent sets of vertices in G . Suppose there are n_A, n_B non-isolated vertices in A and B respectively. Then there is a quantum algorithm that identifies these non-isolated vertices with $O(\sqrt{n_A} + \sqrt{n_B})$ OR queries. The algorithm succeeds with certainty.*

Proof. For each subset $S \subseteq A$, we consider queries of the form $S \cup B$. The result is 1 if and only if there is a non-isolated vertex in S . We can view A as a bit string such that the i -th element is 1 if the i -th vertex is non-isolated, and 0 otherwise. Using the quantum algorithm for CGT (see Theorem 1), we can learn this bit-string with $O(\sqrt{n_A})$ queries. Similarly, we can learn the non-isolated vertices in B with $O(\sqrt{n_B})$ queries. ◀

Note that if there are m_{AB} edges between A, B , then $n_A, n_B \leq \min(m_{AB}, n) \leq \min(m, n)$. Next, we show how to learn the edges between A and B . Lemma 3 below focuses on a general case, and Lemma 4 considers the case of bounded-degree graphs.

► **Lemma 3.** *Make the same assumptions as Lemma 2. Suppose there are m_{AB} edges between A and B . Then there is a quantum algorithm that identifies these edges with $O(m_{AB})$ OR queries. The algorithm succeeds with certainty.*

Proof. By Lemma 2, we assume that there are no isolated vertices in A, B . It costs 1 query to check if $m_{AB} = 0$ or not. In the following, we shall assume that $m_{AB} > 0$. We view each vertex as a variable. Then the learning problem is equivalent to learn the Boolean function $f = x_1 f_1 \vee \dots \vee x_{n_A} f_{n_A}$, where f_1, \dots, f_{n_A} are OR functions of variables $y_1, \dots, y_{n_B} \in B$, and where $x_1, \dots, x_{n_A} \in A$. To learn f , we first set all variables in B to 1, then f becomes $x_1 \vee \dots \vee x_{n_A}$. By the CGT algorithm, we can learn x_1, \dots, x_{n_A} with $O(\sqrt{n_A})$ queries. Next, for each $i \in \{1, \dots, n_A\}$, we set $x_i = 1, x_j = 0$ ($j \neq i$), then we are left with f_i . Using the CGT algorithm again, we can learn f_i with $O(\sqrt{a_i})$ queries, where a_i is the size of f_i , i.e., the number of relevant variables in f_i . Thus the total number of queries is $O(\sqrt{n_A} + \sqrt{a_1} + \dots + \sqrt{a_{n_A}})$. Since $a_1 + \dots + a_{n_A} = m_{AB}$ and $n_A \leq m_{AB}$, the number of queries is bounded by $O(\sqrt{m_{AB}} + m_{AB}) = O(m_{AB})$, which is tight when $a_1 = \dots = a_{n_A} = 1$ and $n_A = m_{AB}$. ◀

When the graph is bounded-degree, the above lemma can be improved.

► **Lemma 4.** *Make the same assumptions as Lemma 3, and additionally suppose that G has maximum degree d . Then there is a quantum algorithm that identifies the edges using $O(d^2 \sqrt{m_{AB}} \log m_{AB})$ OR queries. The algorithm succeeds with certainty.*

Proof. For simplicity, let each vertex in A and B have an index in the set $\{1, \dots, |A|\}, \{1, \dots, |B|\}$ respectively. By Lemma 2, we can assume that $|A| = n_A, |B| = n_B$. That is, there are no isolated vertices in A, B .

First, to gain intuition, we consider the special case of matchings ($d = 1$). In this case, $n_A = n_B = m_{AB} \leq n$. For each $a \in A$, we use n_a to denote the index of the neighbour of a in B , if such a neighbour exists, and otherwise set $n_a = 0$. For any $T \subseteq B$, let $B^T \in \{0, 1\}^{|A|}$ denote the bit-string whose i 'th element equals 1 if $n_i \in T$, and 0 otherwise. Fixing the same T and varying over subsets $S \subseteq A$ and queries of the form $S \cup T$, we can think of this oracle query as returning 1 if there exists $i \in S$ such that $n_i \in T$ (equivalently, $B_i^T = 1$), and 0 otherwise. This is the same oracle used in CGT, so this means that B^T can be learned completely using $O(\sqrt{|B^T|})$ quantum queries for any fixed T . Here $|B^T|$ is the Hamming weight of the bit-string B^T .

We then repeat this algorithm for different choices of T . In particular, we can think of each $n_i \in \{1, \dots, |B|\}$ as an element of $\{0, 1\}^{\lceil \log(|B|+1) \rceil}$, and consider the sequence $T_j = \{i : i_j = 1\}, j = 1, \dots, \lceil \log(|B|+1) \rceil$. Then $k := \lceil \log(|B|+1) \rceil = O(\log m_{AB})$ repetitions are enough to learn all the bits of n_i for all $i \in A$, and hence to learn the graph completely. The overall complexity is $O(\sqrt{|B^{T_1}|} + \dots + \sqrt{|B^{T_k}|})$. As $|B^{T_i}| \leq m_{AB}$ for all i , the complexity is bounded by $O(\sqrt{m_{AB}} \log m_{AB})$. Note that there is no need to repeat the CGT algorithm to reduce its error probability, as it is already exact.

Next, we consider bounded-degree graphs. We can generalise the above idea to learning bipartite graphs where every vertex in A has degree at most d . For each $a \in A$, we now define n_a as the set of the indices of the neighbours of a in B . For any $T \subseteq B$, define $B^T \in \{0, 1\}^{|A|}$ as the bit-string such that the i 'th element equals 1 if $n_i \cap T \neq \emptyset$, and 0 otherwise. Then, for any choice of T , an oracle query of the form $S \cup T$, $S \subseteq A$, returns whether any vertex in S has any neighbours in T . This implies that B^T can be learned with $O(\sqrt{|B^T|})$ queries using the quantum algorithm for CGT [15].

There are randomised constructions of families of subsets T of size $k = O(d^2(\log n_B))$ that allow the d nonzero entries to be determined deterministically, for any pattern of nonzero entries (these are “nonadaptive” combinatorial group testing schemes [3, 10, 39]). Since $|B^{T_i}| \leq m_{AB}$ for all i , the overall complexity is $O(d^2\sqrt{m_{AB}}(\log n_B))$. ◀

There are also nonadaptive combinatorial group testing strategies that are designed to have a low worst-case probability of error [3, 21], and have only a linear dependence on d . However, it is not clear that these schemes can be used in our setting, as the failure probability would be of the form $n_B^{-\delta}$, for some $\delta > 0$, and n_B might be much less than n .

In the following, we consider a more general case when A, B are not independent.

► **Lemma 5.** *Assume that A and B are two disjoint, non-empty sets of vertices in G with m_A, m_B known edges respectively. Suppose there are m_{AB} edges between A and B . Then there is a quantum algorithm that identifies these edges using $O(m_{AB} + m_A + m_B)$ OR queries. In particular, if G has maximal degree d , then the algorithm uses $O(d^4\sqrt{m_{AB}} \log m_{AB})$ queries.*

Proof. The idea behind the quantum algorithm is as follows: We first color the two graphs induced by A, B such that each color class is an independent set in G . Then we use Lemmas 3 and 4 to identify the edges between color classes in A and color classes in B .

It is well-known that a graph with t edges can be $\lfloor \sqrt{2t} + 1 \rfloor$ -colored. The coloring can be constructed in polynomial time. Now let $q_1 = \lfloor \sqrt{2m_A} + 1 \rfloor, q_2 = \lfloor \sqrt{2m_B} + 1 \rfloor$ be the number of colors used for A and B , respectively. Assume that there are m_{ij} edges between the i -th color class of A and the j -th color class of B . Then by Lemma 3, the number of queries used to identify the edges between A and B is bounded by $\sum_{i=1}^{q_1} \sum_{j=1}^{q_2} O(m_{ij} + 1) = O(m_{AB} + q_1 q_2) = O(m_{AB} + m_A + m_B)$.

If G has degree d , then it is d -colorable, namely $q_1, q_2 = O(d)$. By Lemma 4 and the same argument as above, all edges can be identified with $O(d^4\sqrt{m_{AB}} \log m_{AB})$ queries. ◀

The next lemma generalizes the above lemma to learn the edges of multiple disjoint subsets. Note that if there are k subsets, then there are $O(k^2)$ pairs. So naively we need to make at least $O(k^2)$ queries. However, this can be improved to be linear in k by using Lemma 5 in a binary decomposition approach.

► **Lemma 6.** *Assume that S_0, \dots, S_{k-1} are disjoint non-empty sets of vertices in G , and each has s_i known edges. Suppose there are $s_{i,j}$ edges between S_i and S_j , then there is a quantum algorithm that identifies all the edges using $O(k + T \log k)$ OR queries, where $T = \sum_i s_i + \sum_{i,j} s_{i,j}$. If G has maximal degree d , then the number of queries can be reduced to $O(k + d^4\sqrt{kT} \log T)$.*

Proof. For simplicity, we assume that $k = 2^l$ for integer l . Set $K = \sum_{i=0}^{k-1} s_i$. The idea of the algorithm is to recursively use Lemma 5 in a binary form. In step 1, for each pair (S_{2i}, S_{2i+1}) , we use Lemma 5 to find the edges between them. There are 2^{l-1} pairs in total. So this step uses

$$O\left(\sum_{i=0}^{2^{l-1}-1} (s_{2i,2i+1} + s_{2i} + s_{2i+1} + 1)\right) = O\left(2^{l-1} + K + \sum_{i=0}^{2^{l-1}-1} s_{2i,2i+1}\right)$$

queries in total. After step 1, we know the edges of each adjacent pair (S_{2i}, S_{2i+1}) . So we can combine them and obtain a new set, denoted as $S'_i := S_{2i} \cup S_{2i+1}$ for $i = 0, 1, \dots, 2^{l-1} - 1$. It has $s'_i := s_{2i,2i+1} + s_{2i} + s_{2i+1}$ edges. The number of edges between S'_i and S'_j is $s'_{i,j} := s_{2i,2j} + s_{2i,2j+1} + s_{2i+1,2j} + s_{2i+1,2j+1}$. Now, similarly to step 1, we can learn the edges between (S'_{2i}, S'_{2i+1}) . This step uses

$$O\left(2^{l-2} + K + \sum_{i=0}^{2^{l-1}-1} s_{2i,2i+1} + \sum_{i=0}^{2^{l-2}-1} s_{4i,4i+2} + s_{4i,4i+3} + s_{4i+1,4i+2} + s_{4i+1,4i+3}\right)$$

queries in total. Continuing the above procedure, we can learn all the edges. The above procedure terminates after $l = O(\log k)$ steps. It is not hard to show that the total number of queries is bounded by $O(Kl + 2^l + T_1 + T_2 + \dots + T_{l-1})$, where T_i is the total number of edges between two adjacent pairs in step i . Since $T_i \leq T - K$, the number of queries is bounded by $O(T(\log k) + k)$.

When G has maximal degree d , by Lemma 5, the number of queries used in step i is

$$O\left(2^{l-i} + d^4 \sum_{j=0}^{2^{l-i}-1} \sqrt{s_{2j,2j+1}^{(i)}} \log s_{2j,2j+1}^{(i)}\right),$$

where $s_{2j,2j+1}^{(i)}$ is the number of edges of the j -th adjacent pair in step i . It is easy to check that $\sum_{i=1}^{l-1} \sum_{j=0}^{2^{l-i}-1} s_{2j,2j+1}^{(i)} = T - K$, thus the total number of queries used in the algorithm is bounded by $O(k + d^4 \sqrt{kT} \log T)$, where we bound $s_{2j,2j+1}^{(i)} \leq T$ and use Cauchy-Schwarz inequality. \blacktriangleleft

We can now use these ingredients to obtain algorithms for learning general graphs using OR queries. By the above lemma, what remains is to decompose the set of vertices into a disjoint union of a small number of subsets. We shall use the following trick described in [8].

Given a probability p , a p -random set S is obtained by including each vertex independently with probability p . Then the probability that a p -random set includes no edge of G is at least $q = 1 - mp^2$. Choosing $p = 1/10\sqrt{m}$, then the probability is at least $q = 0.99$. The size of S is close to pn with high probability.

Let V denote the set of vertices. First we identify a random set S_1 that includes no edge of G by following the above procedure. After we have S_1 , then in $V - S_1$, we can find another random set S_2 that includes no edge of G . We continue this process for k steps, where k is determined later. Assume now that we have k random sets S_1, \dots, S_k . Each has no edge of G . This uses $O(k)$ queries in total. After k steps, the number of remaining vertices is about $(1-p)^k n \approx e^{-pk} n$. This means that the above procedure terminates after $k = O(p^{-1} \log n) = O(\sqrt{m} \log n)$ steps with high probability.

► Theorem 7. *Let G be a graph with m edges and n vertices. Then there is a quantum algorithm that learns the graph by making*

$$O(m \log(\sqrt{m} \log n) + \sqrt{m} \log n) \tag{1}$$

OR queries with probability at least 0.99. If G has maximal degree d , the query complexity is

$$O(d^4 m^{3/4} \sqrt{\log n} (\log m) + \sqrt{m} \log n) \tag{2}$$

with probability at least 0.99.

Proof. The idea of our algorithm is as follows: we first decompose the vertices of the graph into $k = O(\sqrt{m} \log n)$ independent subsets by the above arguments. Then we learn the edges among all the pairs using Lemma 6.

The first step uses $O(\sqrt{m} \log n)$ OR queries. By Lemma 6, all the edges can be identified with $O(k + m \log k) = O(\sqrt{m}(\log n) + m \log(\sqrt{m} \log n))$ queries. If the graph has maximal degree d , then the number of queries is

$$O\left(d^4 \sqrt{m^{3/2}(\log n)(\log m)} + \sqrt{m} \log n\right) = O\left(d^4 m^{3/4} \sqrt{\log n(\log m)} + \sqrt{m} \log n\right). \quad \blacktriangleleft$$

The quantum query complexity achieved by the first part of Theorem 7 is an improvement over the $\Omega(m \log(n^2/m))$ classical lower bound if m is very small with respect to n ; for example, if $m = \Theta(\log n)$, the complexity is $O(\log^{1.5} n)$, as compared with $\Omega(\log^2 n)$ classically. However, if $m = \Omega(n^\epsilon)$ for some fixed $\epsilon > 0$, the complexity is worse than the classical lower bound.

If G is promised to be a Hamiltonian cycle or a matching (for example), then $d = O(1)$, and by the second part of Theorem 7 the number of OR queries used to learn G is bounded by $O(m^{3/4} \sqrt{\log n(\log m)} + \sqrt{m} \log n)$, which is an improvement over the $\Omega(m \log(n/m))$ classical complexity for large m .

2.1 Learning specific graphs using OR queries

Next we give quantum algorithms for learning some specific graph families using OR queries.

► **Proposition 8.** *There is a quantum algorithm which makes $O(\sqrt{k})$ OR queries and identifies an arbitrary clique on k vertices.*

Proof. The idea is as follows: First, we find a vertex v in the clique, then use the quantum algorithm for CGT [15] to learn all the other vertices using $O(\sqrt{k})$ queries, by querying with subsets of the vertices that include v . Such a query returns 1 if and only if the subset includes another vertex of the clique.

As for the first step, the vertex v can be found with high probability using $O(1)$ queries, using a similar idea to the quantum algorithm of [7] for CGT. We produce a subset S of vertices by including each vertex with probability $1/k$. Then with probability $\binom{k}{2} k^{-2} (1 - 1/k)^{k-2} \approx 1/2e$, this leads to exactly 2 vertices i, j in the clique being included in the subset. This subset corresponds to a boolean function $f(x) = x_i x_j$ for unknown i, j . To learn i, j , we use the Fourier sampling method. Let b_k be the bit-string of length n whose k -th bit equals 1, and all other bits equal 0. It is easy to verify that the Fourier coefficients of f at $b_i, b_j, b_i + b_j$ are all equal to $1/2$. Thus with probability at least $3/4$, we can identify x_i or x_j . ◀

► **Proposition 9.** *There is a quantum algorithm which makes $O(\sqrt{m})$ OR queries and identifies an arbitrary star graph with m edges.*

Proof. This is equivalent to learning the Boolean function $f(x) = x_i \wedge (\bigvee_{j \in A} x_j)$, for some unknown i , A , where A is a subset of $[n]$ of size m and $i \notin A$. To learn it, we use the Fourier sampling to identify the center x_i first, then use the CGT algorithm to learn the edges.

The Fourier sampling method returns a state of the form $\sum_{y \in \{0,1\}^n} \hat{f}(y) |y\rangle$. Consider the Fourier coefficient at $y_i = 1, y_j = 0$ ($j \neq i$). It equals

$$\frac{1}{2^n} \sum_{x \in \{0,1\}^n} (-1)^{x_i \wedge (\bigvee_{j \in A} x_j) + x_i} = \frac{1}{2^n} \left(\sum_{x \in \{0,1\}^n: x_i=0} 1 - \sum_{x \in \{0,1\}^n: x_i=1} (-1)^{\bigvee_{j \in A} x_j} \right) = 1 - \frac{1}{2^{m-1}}.$$

This means that Fourier sampling can detect the center with $O(1)$ queries with high probability. After we obtain the center, it suffices to focus on the function obtained by setting $x_i = 1$. Using the quantum algorithm for CGT, we can learn this function with $O(\sqrt{m})$ queries. ◀

The above two results are tight because of the optimality of CGT. More precisely, CGT corresponds to the special case of learning a clique when one vertex is given, or learning a star when the center is given.

2.2 Lower bound

Finally, we show a quantum lower bound for learning graphs with OR queries, which shows that the quantum algorithm given in Theorem 7 for learning graphs with m edges is optimal up to a logarithmic factor.

► **Theorem 10.** *Let G be an arbitrary graph of n vertices. Then any quantum algorithm that learns G with success probability $> 1/2$ using OR queries must make $\Omega(n^2)$ queries.*

Proof. Consider the family of graphs on $2n$ vertices defined as follows. We first start with two disjoint cliques A, B on n vertices. We then put edges between A and B in an arbitrary pattern. This corresponds to an adjacency matrix of the form $\begin{pmatrix} J - I & M \\ M^T & J - I \end{pmatrix}$, where J is the all-1's matrix, and M is an arbitrary $n \times n$ matrix. Now observe that any query that contains more than one vertex in A , or more than one vertex in B , will always return 1. Any query that contains only one vertex in total will always return 0. So we can restrict to considering queries that include exactly one vertex of A and exactly one vertex of B . Such a query just returns one of the entries of M . Learning M with success probability $> 1/2$ using this oracle requires $\Omega(n^2)$ quantum queries [12]. ◀

As a corollary, we get the lower bound that any quantum algorithm that learns an arbitrary graph with m edges must make $\Omega(m)$ quantum queries. Also, by the known lower bound on the quantum query complexity of the parity function [12], if m is unknown, then any quantum algorithm that determines m exactly must make $\Omega(m)$ queries when $m = \Omega(n^2)$.

3 Learning an unknown graph state

The graph state $|G\rangle$ on n qubits corresponding to a graph $G = (V, E)$ with n vertices can be defined explicitly as

$$|G\rangle = \frac{1}{\sqrt{2^n}} \sum_{x \in \{0,1\}^n} (-1)^{\sum_{(i,j) \in E} x_i x_j} |x\rangle, \quad (3)$$

The state $|G\rangle$ can also be defined as the state produced by acting on the uniform superposition $|+\rangle^{\otimes n}$ with a controlled- Z gate across each pair of qubits corresponding to an edge in G , or as the unique state stabilized by the set of Pauli operators $\{X_v \prod_{w \in N(v)} Z_w : v \in V\}$, where $N(v)$ denotes the set of vertices neighbouring v [32].

The representation (3) makes it clear that graph states have a close connection to the parity query model, as $|G\rangle$ is the state produced by evaluating $f_{G, \text{Par}}(S)$ on all subsets S in uniform superposition. Therefore, lower bounds on the complexity of identifying graphs using parity queries imply lower bounds on the number of copies of $|G\rangle$ required to identify G , and upper bounds on the number of copies of $|G\rangle$ required to identify G imply upper bounds on the complexity of identifying G using parity queries.

1:12 Quantum Algorithms for Learning a Hidden Graph

First we show how to partially go in the other direction, by making parity queries of a certain form, given copies of $|G\rangle$. We use a procedure called Bell sampling, which was used for learning arbitrary stabilizer states in [36]. Given two copies of a state $|\psi\rangle$ of n qubits, Bell sampling corresponds to measuring each corresponding pair of qubits in the Bell basis. Outcomes of Bell sampling can be identified with strings $s \in \{I, X, Y, Z\}^n$ of Pauli matrices, and are observed with the following probabilities:

► **Lemma 11** (Lemma 2 of [36]). *Let $|\psi\rangle$ be a state of n qubits. Bell sampling applied to $|\psi\rangle^{\otimes 2}$ returns outcome s with probability $2^{-n} |\langle \psi | \sigma_s | \psi^* \rangle|^2$, where $|\psi^*\rangle$ is the complex conjugate of $|\psi\rangle$ with respect to the computational basis, and $\sigma_s = s_1 \otimes s_2 \otimes \cdots \otimes s_n$.*

If $|G\rangle$ is a graph state, then $|G\rangle = |G^*\rangle$, and $|\langle G | \sigma_s | G \rangle|^2 = 1$ if and only if σ_s is a stabilizer of $|G\rangle$; otherwise, $|\langle G | \sigma_s | G \rangle|^2 = 0$. Therefore, Bell sampling returns a uniformly random stabilizer of $|G\rangle$. Such a stabilizer can be produced by taking the product of a random subset S of the rows of the stabilizer matrix for G (where each row is included with independent probability $1/2$). We obtain the following overall operator:

$$\prod_{v \in S} X_v \prod_{u \in N(v)} Z_u = \pm \prod_{u \in [n]} X_u^{[u \in S]} Z_u^{N(u) \cap S}$$

where we collect X and Z terms together for each vertex $u \in [n]$. Hence, when we receive a sample of a uniformly random stabilizer of $|G\rangle$, we obtain a random subset $S \subseteq [n]$, and for each $u \in [n]$, we learn the number of edges between u and S , mod 2. We learn the identity of S from which qubits have an X term associated with them.

This allows us to try to find efficient algorithms based only on this (now classical) subroutine of learning subsets and parities. Indeed, learning a graph state using Bell sampling is equivalent to learning a graph using parity queries, as studied in Section 4 below – except with the restriction that these queries are only to uniformly random subsets of the vertices. We first give a general algorithm for learning a graph known to be picked from any finite set.

► **Theorem 12.** *Let S be a family of graphs. Then, for any $G \in S$, G can be identified by applying Bell sampling to $O(\log |S|)$ copies of $|G\rangle$. The algorithm succeeds with probability at least 0.99.*

Proof. Let A be the adjacency matrix of G . Each Bell sample returns the inner product of a random vector $\mathbf{s} \in \mathbb{F}_2^n$ with each column (or row) of A . If we take k samples, we can write these k row vectors as an $n \times k$ matrix B . Then the result of the Bell sampling procedure is the matrix AB .

To be able to uniquely identify G , we want $AB \neq A'B$ for all A, A' corresponding to graphs in S , or in other words $(A + A')B \neq 0^{n \times k}$. As each entry of B is uniformly random, for any $n \times n$ matrix C with rank r , $\Pr_B[CB = 0^{n \times k}] = 2^{-kr}$. (This holds because for each linearly independent row \mathbf{c} of C , $\Pr_B[\mathbf{c}B = 0^k] = 2^{-k}$, and these events are independent.) In particular, for any nonzero matrix C , $\Pr_B[CB = 0^{n \times k}] \leq 2^{-k}$. The number of matrices C of the form $C = A + A'$ is at most $|S|^2$. Taking a union bound over all such matrices, we have

$$\Pr_B[\exists C = A + A', CB = 0^{n \times k}] \leq \frac{|S|^2}{2^k}.$$

So it is sufficient to take $k = O(\log |S|)$ to achieve failure probability 0.01, as claimed. ◀

As a corollary of Theorem 12, if G is a graph with at most m edges, it can be identified with $O(m \log(n^2/m))$ copies of $|G\rangle$.

It is natural to wonder whether the dependence on $|S|$ in Theorem 12 could be improved, because if S is the set of all graphs, the complexity of Theorem 12 does not match that of the best algorithms for learning an arbitrary graph state, which use $O(n)$ copies of $|G\rangle$ [1, 36, 42]. An information-theoretic lower bound comes from the fact that $|G\rangle$ is a state of n qubits, so by Holevo's theorem, $\Omega((\log |S|)/n)$ copies are required to identify a state from S . In addition, this bound cannot always be reached; if S is the set of all graphs on r vertices, for some $r < n$, the number of copies required to identify a graph from this set is $\Theta(r)$ by the same information-theoretic argument, which can be much larger than $O(r^2/n)$ for some choices of r . This suggests that the best dependence on $|S|$ that could be achieved is $O(\sqrt{\log |S|})$.

However, better complexities can be achieved for graphs with more structure. If the graph is promised to be a star, then the Fourier sampling method can be applied to learn it with $O(1)$ copies of $|G\rangle$. More precisely, suppose the edges of the star graph are $(i, j), j \in A$. Here i is the center and we assume $|A| \geq 1$. Then

$$|G\rangle = \frac{1}{\sqrt{2^n}} \sum_{x \in \{0,1\}^n} (-1)^{x_i \sum_{j \in A} x_j} |x\rangle.$$

By Fourier sampling, if we apply Hadamard gates to $|G\rangle$, we obtain the state

$$\frac{1}{\sqrt{2}} |0, \dots, 0\rangle |+\rangle |0, \dots, 0\rangle + \frac{1}{\sqrt{2}} |[1 \in A], \dots, [i-1 \in A]\rangle |-\rangle |[i+1 \in A], \dots, [n \in A]\rangle. \quad (4)$$

The $|\pm\rangle$ is in the i -th qubit. Performing measurements in the computational basis, if we obtain $|0, \dots, 0\rangle |1\rangle |0, \dots, 0\rangle$, then we know the center; if we obtain a state with more than two 1's, then we know all vertices in A . The probability is $1/4$ for each case, so we can learn a unknown star using $O(1)$ copies of $|G\rangle$.

We can also apply Bell sampling to learn cliques with $O(1)$ copies. Each Bell sample gives us the inner product of each row of the adjacency matrix with a random vector, and each nonzero row has probability $1/2$ for this inner product to be nonzero. As G is a clique, all its nonzero rows are the same. Thus, after $O(1)$ samples, with high probability we learn all the nonzero rows at once.

In summary, we have

► **Theorem 13.** *There is a quantum algorithm that identifies G by using $O(1)$ copies of $|G\rangle$ if G is a star or a clique.*

Next we consider the case of bounded-degree graphs.

► **Theorem 14.** *For an arbitrary graph G , there is a quantum algorithm which uses $O(d \log(m/d))$ copies of $|G\rangle$, and for each vertex v that has degree at most d , outputs all the neighbours of v and that v has degree at most d . For each vertex w that has degree larger than d , the algorithm outputs “degree larger than d ”. The algorithm succeeds with probability at least 0.99 .*

Proof. We assume that $d \leq n/4$ throughout, as otherwise an algorithm for learning an arbitrary graph using $O(n)$ copies can be used [1, 36, 42]. We produce k Bell samples, corresponding to vectors $A\mathbf{w}_1, \dots, A\mathbf{w}_k$ for uniformly random vectors $\mathbf{w}_1, \dots, \mathbf{w}_k \in \{0,1\}^n$. For any pair $\mathbf{x} \neq \mathbf{y} \in \{0,1\}^n$, the probability that $\mathbf{x} \cdot \mathbf{w}_i = \mathbf{y} \cdot \mathbf{w}_i$ for all i is equal to the probability that $(\mathbf{x} + \mathbf{y}) \cdot \mathbf{w}_i = 0$ for all i , which equals 2^{-k} . By a union bound, for any $\mathbf{x} \in \{0,1\}^n$, the probability that there exists $\mathbf{y} \in \{0,1\}^n$ such that $\mathbf{y} \neq \mathbf{x}$, $|\mathbf{y}| \leq d$ and $\mathbf{x} \cdot \mathbf{w}_i = \mathbf{y} \cdot \mathbf{w}_i$ for all i is bounded by $\sum_{l=0}^d \binom{n}{l} 2^{-k} = O(2^{d \log(n/d) - k})$.

We then apply this bound to all n rows of A via a union bound, to obtain that the probability that, for any row \mathbf{x} of A , there exists $\mathbf{y} \in \{0, 1\}^n$ with $|\mathbf{y}| \leq d$, $\mathbf{x} \cdot \mathbf{w}_i = \mathbf{y} \cdot \mathbf{w}_i$ for all i and $\mathbf{y} \neq \mathbf{x}$ is $O(n2^{d \log(n/d) - k})$. Taking $k = O(d \log(n/d))$ is sufficient to bound this probability by an arbitrarily small constant. Assuming that this failure event does not occur, the algorithm determines all rows of A with Hamming weight bounded by d , and identifies all rows that are inconsistent with having Hamming weight bounded by d .

We finally show how to replace n with m in the algorithm's complexity. This is achieved by first identifying the subset W of non-isolated vertices, and then running the algorithm above on the vertices in this subset. We can restrict the graph to this subgraph H by measuring the qubits corresponding to the other vertices in the computational basis. The resulting state is of the form $|H'\rangle = \prod_{i \in T} Z_i |H\rangle$, for some subset $T \subseteq W$. By Lemma 11, Bell sampling behaves in the same way on $|H'\rangle$ as on $|H\rangle$. To find the subset W , Bell sampling is applied l times for some l , to produce an $n \times l$ matrix $C = AB$ for a uniformly random matrix B . The set of vertices corresponding to rows of C which have at least one nonzero entry is kept, to produce a set W' . Any zero row of A will always produce a corresponding zero row of C , so will not be included in W' . On the other hand, the probability that any nonzero row of A produces the corresponding zero row of C is 2^{-l} . As there are at most $2m$ nonzero rows, corresponding to vertices in W , the probability that any vertex in W is not included in W' is $O(m2^{-l})$ by a union bound. So it is sufficient to take $l = O(\log m)$ to learn which rows are nonzero with probability 0.99. ◀

We can also learn the family of graphs that are subgraphs of a fixed graph G' of bounded degree d . This is relevant to the setting where we have attempted to produce $|G'\rangle$ using a quantum circuit which may have failed to produce certain edges, and we would like to determine which graph we have actually produced. In this case, we can get an algorithm that still uses $O(d \log n)$ copies like Theorem 14, but is also computationally efficient, in that its runtime is $O(d^3 n \log^3 n)$.

► **Theorem 15.** *Let G' be a graph of bounded degree d , G be a subgraph of G' . Given access to copies of $|G\rangle$, there is a quantum algorithm that identifies G using $O(d \log n)$ copies with runtime $O(d^3 n \log^3 n)$. The algorithm succeeds with probability at least 0.99.*

Proof. We take k Bell samples, for some k to be determined. For each vertex v , the corresponding row \mathbf{r}_v of A is a linear combination over \mathbb{F}_2 of at most d fixed vectors $\mathbf{e}_1, \dots, \mathbf{e}_d$ of Hamming weight 1, where each vector corresponds to a neighbour of v in G' . So we can write $\mathbf{r}_v = \sum_{i=1}^d x_i \mathbf{e}_i$ for some $x_i \in \{0, 1\}$, and determining $\mathbf{x} \in \{0, 1\}^d$ is sufficient to determine \mathbf{r}_v . As the results of the Bell samples correspond to inner products between \mathbf{r}_v and random vectors over \mathbb{F}_2^n , we obtain a system of k random linear equations in d unknowns. These equations can be solved in time $O(k^3)$ to determine \mathbf{x} if the corresponding random matrix is full rank, and the probability that a random $k \times d$ matrix over \mathbb{F}_2 , $k \geq d$, is not full rank is $O(2^{-(k-d)})$ [28]. So, by a union bound, it is sufficient to take $k = O(d \log n)$ for all of the rows of A to be determined by solving the corresponding systems of linear equations. ◀

Using a similar technique to the last part of Theorem 14, the linear dependence on n in Theorem 15 can be replaced with a linear dependence on the number of non-isolated vertices, and the polylog dependence on n can be replaced with an equivalent dependence on m .

4 Learning an unknown graph with parity queries

In this section we investigate learning an unknown graph G using the parity oracle $f_{G,\text{Par}}(S)$. Identifying S with a bit-string $x \in \{0, 1\}^n$ via $x_i = 1$ if $i \in S$, and $x_i = 0$ otherwise, we see that $f_{G,\text{Par}}(x) = \sum_{(i,j) \in E} x_i x_j$, where the sum is taken mod 2. So, if G is arbitrary, $f_{G,\text{Par}}$ is an arbitrary quadratic polynomial over \mathbb{F}_2 with no linear part. It was shown in [35] that any polynomial of this form can be learned using $O(n)$ quantum queries, and this is optimal. This immediately gives a quantum algorithm for learning an arbitrary graph using $O(n)$ parity queries, which is quadratically better than the best possible classical algorithm. (By an information-theoretic argument, classically $\Omega(n^2)$ parity queries are required.)

Evaluating $f_{G,\text{Par}}(x)$ on a uniform superposition over computational basis states $|x\rangle$ gives precisely the graph state $|G\rangle$, so the results of Section 3 can all immediately be applied to learning graphs in the parity query model. However, the ability to evaluate $f_{G,\text{Par}}(x)$ on other input states allows for more general algorithms to be developed. In particular, we can obtain the following subroutine.

► **Lemma 16.** *Let A be the adjacency matrix of G . For any $v \in \{0, 1\}^n$, there is a quantum algorithm which returns Av and makes two queries to $f_{G,\text{Par}}$.*

Proof. Consider the function $g_v(x) = f(x) + f(x+v)$. It can be evaluated for any x using two queries to f . Let B denote the adjacency matrix A , except that we set $B_{ij} = 0$ for $i > j$. Then $f(x) = x^T Bx$.

We evaluate g_v in superposition to produce

$$\begin{aligned} \frac{1}{\sqrt{2^n}} \sum_{x \in \{0,1\}^n} (-1)^{g_v(x)} |x\rangle &= \frac{1}{\sqrt{2^n}} \sum_{x \in \{0,1\}^n} (-1)^{x^T Bx + (x+v)^T B(x+v)} |x\rangle \\ &= \frac{1}{\sqrt{2^n}} (-1)^{v^T Bv} \sum_{x \in \{0,1\}^n} (-1)^{x \cdot (Av)} |x\rangle. \end{aligned}$$

Then applying Hadamard gates to each qubit returns the vector Av . ◀

Note that no equivalent of Lemma 16 can hold in the graph state model of Section 3. If we let v be a vector of Hamming weight 1, Lemma 16 returns an entire row of the adjacency matrix of A using one query. But even to determine one entry of an arbitrary row of A requires $\Omega(n)$ copies of $|G\rangle$, because this is equivalent to a quantum random access code on $\binom{n}{2}$ bits¹. Such codes are known to require quantum states of $\Omega(n^2)$ qubits [37], and $|G\rangle$ is a state of n qubits.

We can use Lemma 16 as a subroutine to learn an arbitrary graph with a bounded number of edges. Classically, by an information-theoretic argument, this requires at least $\Omega(\log \binom{n}{m}) = \Omega(m \log(n^2/m))$ queries.

► **Theorem 17.** *There is a quantum algorithm which learns a graph with at most m edges using $O(\sqrt{m \log m})$ parity queries. The algorithm succeeds with probability at least 0.99.*

Proof. The algorithm splits the graph into low and high-degree parts. First, Theorem 14 is used with $d = \sqrt{m/\log m}$. This learns all rows of A with at most $\sqrt{m/\log m}$ nonzero entries, and the identities of all “dense” rows of A with more than $\sqrt{m/\log m}$ nonzero entries. Then each of the dense rows is learned individually by applying Lemma 16 with v chosen to be the corresponding standard basis vector. There can be at most $O(\sqrt{m \log m})$ dense rows, so the overall algorithm uses $O(\sqrt{m \log m})$ queries. ◀

¹ Joe Fitzsimons, personal communication.

Theorem 17 is close to tight, because identifying an arbitrary graph on k vertices (and hence with up to $\Theta(k^2)$ edges) requires $\Omega(k)$ quantum queries [35]. Stars and cliques can be learned with $O(1)$ parity queries via the techniques of the previous section for graph states.

References

- 1 Scott Aaronson and Daniel Gottesman. Identifying stabilizer states, 2008. URL: <http://pirsa.org/08080052/>.
- 2 Hasan Abasi and Bshouty Nader. On learning graphs with edge-detecting queries. In *Algorithmic Learning Theory*, pages 3–30. PMLR, 2019. [arXiv:1803.10639](https://arxiv.org/abs/1803.10639).
- 3 M. Aldridge, O. Johnson, and J. Scarlett. Group testing: An information theory perspective. *Foundations and Trends in Communications and Information Theory*, 15(3–4):196–392, 2019. [arXiv:1902.06002](https://arxiv.org/abs/1902.06002).
- 4 Noga Alon and Vera Asodi. Learning a hidden subgraph. *SIAM Journal on Discrete Mathematics*, 18(4):697–712, 2005. [doi:10.1137/S0895480103431071](https://doi.org/10.1137/S0895480103431071).
- 5 Noga Alon, Richard Beigel, Simon Kasif, Steven Rudich, and Benny Sudakov. Learning a hidden matching. *SIAM Journal on Computing*, 33(2):487–501, 2004. [doi:10.1137/S0097539702420139](https://doi.org/10.1137/S0097539702420139).
- 6 Andris Ambainis, Aleksandrs Belovs, Oded Regev, and Ronald de Wolf. Efficient quantum algorithms for (gapped) group testing and junta testing. In *Proc. 27th ACM-SIAM Symp. Discrete Algorithms*, pages 903–922, 2016. [doi:10.1137/1.9781611974331.ch65](https://doi.org/10.1137/1.9781611974331.ch65).
- 7 Andris Ambainis and Ashley Montanaro. Quantum algorithms for search with wildcards and combinatorial group testing. *Quantum Information and Computation*, 14(5&6), 2014. [doi:10.26421/QIC14.5-6](https://doi.org/10.26421/QIC14.5-6).
- 8 Dana Angluin and Jiang Chen. Learning a hidden hypergraph. *Journal of Machine Learning Research*, 7:2215–2236, 2006.
- 9 Dana Angluin and Jiang Chen. Learning a hidden graph using $O(\log n)$ queries per edge. *Journal of Computer and System Sciences*, 74(4):546–556, 2008. [doi:10.1016/j.jcss.2007.06.006](https://doi.org/10.1016/j.jcss.2007.06.006).
- 10 George K Atia and Venkatesh Saligrama. Boolean compressed sensing and noisy group testing. *IEEE Transactions on Information Theory*, 58(3):1880–1901, 2012. [arXiv:0907.1061](https://arxiv.org/abs/0907.1061).
- 11 A. Atıcı and R. Servedio. Improved bounds on quantum learning algorithms. *Quantum Information Processing*, 4(5):355–386, 2005. [quant-ph/0411140](https://arxiv.org/abs/quant-ph/0411140).
- 12 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](https://doi.org/10.1145/502090.502097).
- 13 Paul Beame, Sariel Har-Peled, Ramamoorthy S. Natarajan, Cyrus Rashtchian, and Makrand Sinha. Edge estimation with independent set oracles. *ACM Transactions on Algorithms (TALG)*, 16(4):1–27, 2020. [doi:10.1145/3404867](https://doi.org/10.1145/3404867).
- 14 Richard Beigel, Noga Alon, Simon Kasif, Mehmet S. Apaydin, and Lance Fortnow. An optimal procedure for gap closing in whole genome shotgun sequencing. In *RECOMB 2001*, pages 22–30, 2001. [doi:10.1145/369133.369152](https://doi.org/10.1145/369133.369152).
- 15 Aleksandrs Belovs. Quantum algorithms for learning symmetric juntas via the adversary bound. *Computational Complexity*, 24:255–293, 2015. [doi:10.1007/s00037-015-0099-2](https://doi.org/10.1007/s00037-015-0099-2).
- 16 Shalev Ben-David, Andrew M. Childs, András Gilyén, William Kretschmer, Supartha Podder, and Daochen Wang. Symmetries, graph properties, and quantum speedups, 2020. [arXiv:2006.12760](https://arxiv.org/abs/2006.12760).
- 17 Ethan Bernstein and Umesh Vazirani. Quantum complexity theory. *SIAM Journal on Computing*, 26(5):1411–1473, 1997. [doi:10.1137/S0097539796300921](https://doi.org/10.1137/S0097539796300921).
- 18 Mathilde Bouvel, Vladimir Grebinski, and Gregory Kucherov. Combinatorial search on graphs motivated by bioinformatics applications: a brief survey. In *WG 2005: Graph-Theoretic Concepts in Computer Science*, pages 16–27, 2005. [doi:10.1007/11604686_2](https://doi.org/10.1007/11604686_2).

- 19 G. Brassard, P. Høyer, M. Mosca, and A. Tapp. Quantum amplitude amplification and estimation. *Quantum Computation and Quantum Information: A Millennium Volume*, pages 53–74, 2002. [quant-ph/0005055](#).
- 20 Nader H Bshouty and Hanna Mazzawi. Reconstructing weighted graphs with minimal query complexity. *Theoretical computer science*, 412(19):1782–1790, 2011. [doi:10.1016/j.tcs.2010.12.055](#).
- 21 Chun Lam Chan, Pak Hou Che, Sidharth Jaggi, and Venkatesh Saligrama. Non-adaptive probabilistic group testing with noisy measurements: Near-optimal bounds with efficient algorithms. In *2011 49th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, pages 1832–1839. IEEE, 2011. [arXiv:1107.4540](#).
- 22 Huilan Chang, Hong-Bin Chen, Hung-Lin Fu, and Chie-Huai Shi. Reconstruction of hidden graphs and threshold group testing. *Journal of Combinatorial Optimization*, 22:270–281, 2011. [doi:10.1007/s10878-010-9291-0](#).
- 23 Huilan Chang, Hung-Lin Fu, and Chih-Huai Shih. Learning a hidden graph. *Optimization Letters*, 8:2341–2348, 2014. [doi:10.1007/s11590-014-0751-9](#).
- 24 H.-B. Chen and H.-L. Fu. Nonadaptive algorithms for threshold group testing. *Discrete Applied Mathematics*, 157:1581–1585, 2009.
- 25 Sung-Soon Choi and Jeong H. Kim. Optimal query complexity bounds for finding graphs. *Artificial Intelligence*, 174(9-10):551–569, 2010. [doi:10.1016/j.artint.2010.02.003](#).
- 26 Dingzhu Du, Frank K Hwang, and Frank Hwang. *Combinatorial group testing and its applications*, volume 12. World Scientific, 2000.
- 27 Christoph Dürr, Mark Heiligman, Peter Høyer, and Mehdi Mhalla. Quantum query complexity of some graph problems. *SIAM Journal on Computing*, 35(6):1310–1328, 2006. [doi:10.1137/050644719](#).
- 28 Paulo JSG Ferreira, Bruno Jesus, Jose Vieira, and Armando J Pinho. The rank of random binary matrices and distributed storage applications. *IEEE communications letters*, 17(1):151–154, 2012.
- 29 Vladimir Grebinski and Gregory Kucherov. Optimal query bounds for reconstructing a Hamiltonian cycle in complete graphs. In *Fifth Israel Symposium on the Theory of Computing Systems*, pages 166–173, 1997. [doi:10.1109/ISTCS.1997.595169](#).
- 30 Vladimir Grebinski and Gregory Kucherov. Reconstructing a hamiltonian cycle by querying the graph: Application to dna physical mapping. *Discrete Applied Mathematics*, 88(1-3):147–165, 1998. [doi:10.1016/S0166-218X\(98\)00070-5](#).
- 31 Vladimir Grebinski and Gregory Kucherov. Optimal reconstruction of graphs under the additive model. *Algorithmica*, 28(1):104–124, 2000. [doi:10.1007/s004530010033](#).
- 32 Marc Hein, Wolfgang Dür, Jens Eisert, Robert Raussendorf, Maarten Van den Nest, and Hans J. Briegel. Entanglement in graph states and its applications. In *Quantum Computers, Algorithms and Chaos, International School of Physics, Enrico Fermi*. IOS Press, 2006. [doi:10.3254/978-1-61499-018-5-115](#).
- 33 I. Krasikov and S. Litsyn. Survey of binary Krawtchouk polynomials. In *Codes and Association Schemes*, volume 56 of *DIMACS series in Discrete Mathematics and Theoretical Computer Science*, pages 199–212. American Mathematical Society, 1999.
- 34 Troy Lee, Miklos Santha, and Shengyu Zhang. Quantum algorithms for graph problems with cut queries. In *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 939–958. SIAM, 2021. [doi:10.1137/1.9781611976465.59](#).
- 35 Ashley Montanaro. The quantum query complexity of learning multilinear polynomials. *Information Processing Letters*, 112(11):438–442, 2012. [doi:10.1016/j.ipl.2012.03.002](#).
- 36 Ashley Montanaro. Learning stabilizer states by Bell sampling, 2017. [arXiv:1707.04012](#).
- 37 Ashwin Nayak. Optimal lower bounds for quantum automata and random access codes. In *Proc. 40th Annual Symp. Foundations of Computer Science*, pages 369–376, 1999. [doi:10.1109/SFFCS.1999.814608](#).

- 38 R. O’Donnell. *Computational Applications Of Noise Sensitivity*. PhD thesis, Carnegie Mellon University, 2003.
- 39 Ely Porat and Amir Rothschild. Explicit non-adaptive combinatorial group testing schemes. In *International Colloquium on Automata, Languages, and Programming*, pages 748–759. Springer, 2008. doi:10.1007/978-3-540-70575-8_61.
- 40 Lev Reyzin and Nikhil Srivastava. Learning and verifying graphs using queries with a focus on edge counting. In *International Conference on Algorithmic Learning Theory*, pages 285–297. Springer, 2007. doi:10.1007/978-3-540-75225-7_24.
- 41 Rocco A. Servedio and Steven J. Gortler. Equivalences and separations between quantum and classical learnability. *SIAM Journal on Computing*, 33(5):1067–1092, 2004. doi:10.1137/S0097539704412910.
- 42 Liming Zhao, Carlos A. Pérez-Delgado, and Joseph F. Fitzsimons. Fast graph operations in quantum computation. *Physical Review A*, 93:032314, 2016. doi:10.1103/PhysRevA.93.032314.

A Combinatorial group testing

In this appendix, we move on from the problem of learning graphs to combinatorial group testing (CGT). In the CGT problem, we are given oracle access to an n -bit string A with Hamming weight at most k . Usually, we assume that $k \ll n$. In one query, we can get the OR of an arbitrary subset of the bits of A . The goal is to determine A using the minimal number of queries. (To connect to the topic of the previous sections, we can see CGT as the problem of learning a graph on n vertices with OR queries, in the very special case where the graph is promised to have no edges between vertices, and may contain up to k self-loops.)

We can think of A as a subset of $[n]$, and define the oracle as

$$f_A(S) = \begin{cases} 1, & \text{if } A \cap S \neq \emptyset, \\ 0, & \text{otherwise.} \end{cases} \quad (5)$$

Classically, it is known that the number of queries required to solve CGT is $\Theta(k \log(n/k))$ [26]. In the quantum case, Ambainis and Montanaro [7] first studied this problem and proposed a quantum algorithm using $O(k)$ queries. They also showed a lower bound of $\Omega(\sqrt{k})$. Later in [15], based on the adversary bound method, Belovs proved that a quantum computer can solve the CGT problem with $\Theta(\sqrt{k})$ queries. In principle, Belovs’ approach can yield a quantum algorithm with an explicit implementation, but this implementation might not be time-efficient. In this section, we propose a quantum algorithm for CGT with an efficient implementation. The complexity is a little worse than $\Theta(\sqrt{k})$ by a factor of $O((\log k)(\log \log k))$.

The idea of our quantum algorithm is inspired by [6] and the Bernstein–Vazirani algorithm [17]. The key idea is to observe that the Bernstein–Vazirani algorithm allows the identity of a subset $A \subseteq [n]$ to be determined with one query to an oracle that computes $|A \cap T|$ for arbitrary $T \subseteq [n]$. And in [6], Ambainis et al solved a closely related problem to evaluating this oracle, which they called gapped group testing (GGT): given the oracle f_A , decide if $|A| \leq k$ or $|A| \geq k + d$. They showed that $\Theta(\sqrt{k/d})$ queries are enough to solve this problem by the adversary bound method. The main idea of their quantum algorithm was borrowed from [15], but unlike [15], they have an efficient implementation of their quantum algorithm.

So it seems that, by taking $d = 1$ and using binary search, we can use the quantum algorithm of [6] for the gapped group testing problem to determine $|A|$ with $O(\log k)$ repetitions of their algorithm, leading to a query complexity of $O(\sqrt{k} \log k)$. However,

we should be careful at this point since the quantum algorithm of [6] only succeeds with probability $2/3$. So $O(\log k)$ repetitions will decrease the success probability to almost 0. A simple method to increase the success probability to $1 - O((\log k)^{-1})$ is using the Chernoff bound. We can think of the intended output of the algorithm of [6] for GGT as 1 if $|A| \leq k$ and 0 if $|A| \geq k+1$. Denote this outcome O . As proved in [6], the probability that each run of the algorithm returns the intended outcome is at least $2/3$. We repeat the algorithm for GGT t times and output the median of the results. Let X be the median. Then by the Chernoff bound, we have $\Pr[X \neq O] \leq e^{-ct}$ for some constant c . So by choosing $t = O(\log \log k)$, the success probability is increased to $1 - c'(\log k)^{-1}$ for an arbitrarily small constant c' . Taking a union bound over the $\lceil \log_2 k \rceil$ uses of the algorithm, we can determine $|A|$ with success probability $9/10$. By applying this algorithm to subsets $S \subseteq [n]$, for varying subsets $|S|$, we can determine $|A \cap S|$ with success probability $9/10$.

Next we show that access to an oracle of this form is sufficient to determine A completely. In fact, this claim holds for any monotone function, rather than just the OR function.

► **Lemma 18.** *Consider a family of monotone boolean functions $g : \{0,1\}^k \rightarrow \{0,1\}$. Assume there is a family of classical or quantum algorithms \mathcal{A}_n which, when applied to $f : \{0,1\}^n \rightarrow \{0,1\}$ such that $f(x) = g(x_S)$ for some subset S such that $|S| = k$, outputs k with success probability $9/10$. Let $T(n)$ denote the complexity of \mathcal{A}_n , and assume that $T(n)$ is nondecreasing. Then there is a quantum algorithm which determines S with success probability $1 - \delta$, for any $\delta > 0$, and has complexity $O(T(n) \log 1/\delta)$.*

Proof. Identify n -bit strings with subsets of $[n]$, and create the uniform superposition $\frac{1}{\sqrt{2^n}} \sum_{T \subseteq [n]} |T\rangle$. For each T , run $\mathcal{A}_{|T|}$ on the function $f_T : \{0,1\}^{|T|} \rightarrow \{0,1\}$ given by f restricted to the variables in T . As f is monotone, a query to f_T can be simulated by a query to f by setting the variables outside of T to 0. The result is a state of the form

$$\frac{1}{\sqrt{2^n}} \sum_{T \subseteq [n]} |T\rangle (\sqrt{1 - \delta_T} |S \cap T\rangle + \sqrt{\delta_T} |\psi_T\rangle)$$

for some $\delta_T \in [0,1]$ such that $\delta_T \leq 1/3$, and some states $|\psi_T\rangle$ such that $\langle S \cap T | \psi_T \rangle = 0$. Apply $Z^{\otimes |T|}$ to the last register and uncompute $\mathcal{A}_{|T|}$ to produce

$$\frac{1}{\sqrt{2^n}} \sum_{T \subseteq [n]} (-1)^{|S \cap T|} (1 - \delta_T) |T\rangle |0\rangle + |\eta\rangle$$

for some unnormalised state $|\eta\rangle$ orthogonal to $|0\rangle$ on the second register. Measure the second register and output “fail” if the result is not 0. Otherwise, apply Hadamard gates to every qubit of the remaining register, and return the result.

The algorithm outputs failure with probability $1 - \frac{1}{2^n} \sum_T (1 - \delta_T)^2 \leq 2\delta_T - \delta_T^2 \leq 1/5$. If the algorithm does not output failure, the residual state has squared inner product $(\frac{1}{2^n} \sum_T (1 - \delta_T))^2 \geq (9/10)^2$ with the state $\frac{1}{\sqrt{2^n}} \sum_{T \subseteq [n]} (-1)^{|S \cap T|} |T\rangle$; if applied to this state, it would output S with certainty, by the analysis of the Bernstein-Vazirani algorithm. Therefore the algorithm fails with probability at most $1/5 + 19/100 < 1/2$. Repetition and taking the majority vote reduces the failure probability to δ , for arbitrary $\delta > 0$, with an additional multiplicative cost $O(\log 1/\delta)$. ◀

By Lemma 18, we obtain the following theorem.

► **Theorem 19.** *There is a quantum algorithm that solves the CGT problem with success probability at least $2/3$. The query complexity is $O(\sqrt{k}(\log k)(\log \log k))$, and time complexity is $\tilde{O}(n\sqrt{k})$.*

B Majority and exact-half functions

In this part, we consider the following general learning problem. We are given access to a function $f : \{0, 1\}^n \rightarrow \{0, 1\}$, which is promised to be equal to some known function $g : \{0, 1\}^k \rightarrow \{0, 1\}$ acting on a subset S of the variables. Our goal is to learn which k variables f depends on. We first note that, for any function g , any classical algorithm for this problem must make $\Omega(\log \binom{n}{k}) = \Omega(k \log(n/k))$ queries, as each query returns 1 bit of information. For some functions g , quantum algorithms can do better. In particular, using the adversary bound method, Belovs [15] showed that for the exact-half function ($g(x) = 1 \Leftrightarrow |x| = k/2$) and the majority function ($g(x) = 1 \Leftrightarrow |x| \geq k/2$), the quantum query complexity of identifying S is $O(k^{1/4})$. Here we give simple explicit quantum algorithms that match this complexity up to logarithmic factors. Then we observe that an even simpler approach can be used to solve this learning problem for almost all functions g .

The approach used in this section is based on applying Fourier sampling to f (see Section 1.3), an approach explored by Atıcı and Servedio [11] in the context of quantum learning and testing algorithms for functions with few relevant variables. Fourier sampling allows one to produce the state $|\psi_f\rangle = \sum_{T \subseteq [n]} \widehat{f}(T) |T\rangle$ with one quantum query to f . Now observe that, if $f(x)$ does not depend on the i 'th bit x_i , $\widehat{f}(T) = 0$ for all T such that $i \in T$. So, if f depends only on a subset S of the variables, measuring $|\psi_f\rangle$ in the computational basis returns a subset of S . By repeating this procedure we can hope to learn all of S , and we can sometimes accelerate this process using amplitude amplification. Let $W_l(g)$ be the Fourier weight of g on the l 'th level, $W_l(g) = \sum_{T, |T|=l} \widehat{g}(T)^2$. Similarly define $W_{\geq l}(g) = \sum_{T, |T| \geq l} \widehat{g}(T)^2$.

► **Lemma 20.** *Let g be a symmetric function, i.e. $g(x) = h(|x|)$ for some h , where $|x|$ is the Hamming weight of x . Then, for any l such that $W_{\geq l}(g) > 0$, there is a quantum algorithm which identifies S with probability at least 0.99 using $O(k/(l\sqrt{W_{\geq l}(g)})) \log k$ queries to f . If $l = k$, there is a quantum algorithm using $O(1/\sqrt{W_k(g)})$ queries.*

Proof. We start by applying amplitude amplification [19] to the following procedure: use Fourier sampling on f , and return “yes” if the size of the subset returned is at least l . This returns a subset T of size $|T| \geq l$ using $O(1/\sqrt{W_{\geq l}(g)})$ evaluations of f and with success probability $\max\{1 - W_{\geq l}(g), W_{\geq l}(g)\} \geq 1/2$ [19, Theorem 2]. Observe that, as $|\psi_f\rangle$ has no support on subsets that are not contained within S , $T \subseteq S$ with certainty.

As g is symmetric, $\widehat{f}(T)$ depends only on $|T|$ for all T , so T is picked uniformly at random from all l '-subsets of $[k]$. For any r , it is sufficient to perform this procedure $O(r)$ times to achieve r successes with high probability. The final step of the algorithm is to output the union of the subsets returned in successful iterations. By a union bound, the probability that there is a variable that is not included in any of the subsets is at most $k(1 - l/k)^r \leq ke^{-lr/k+r}$. So it is sufficient to take $r = O((k/l) \log(k/\delta))$ to achieve success probability $1 - \delta$. For the second claim in the lemma, if $l = k$, we learn all the relevant variables with one use of amplitude amplification and with probability $\geq 1/2$, which can be boosted to arbitrarily close to 1 with a constant number of repetitions. ◀

Lemma 20 crucially relies on g being symmetric. Otherwise, certain variables could be substantially harder to identify than others. To apply Lemma 20, it is sufficient to find bounds on the Fourier spectrum of g , which we now obtain for certain functions. First, we consider the majority function ($\text{MAJ}_k(x) = 1 \Leftrightarrow |x| \geq k/2$), which is a special case of a previously studied framework known as “threshold group testing” [24].

► **Fact 21** ([38], Theorem 3.5.3). *Let MAJ_k be the majority function on k bits. If $|S|$ is even, then $\widehat{MAJ}_k(S) = 0$. Otherwise,*

$$\widehat{MAJ}_k(S) = (-1)^{(k-1)/2} \frac{\binom{(k-1)/2}{(|S|-1)/2}}{\binom{k-1}{|S|-1}} \frac{2}{2^k} \binom{k-1}{(k-1)/2}.$$

Using Fact 21, we can obtain a bound on the tail of the Fourier spectrum of the majority function.

► **Lemma 22.** $W_{\geq (k+1)/2}(MAJ_k) = \Omega(1/\sqrt{k})$.

Proof. By Fact 21,

$$W_l(MAJ_k) = \binom{k}{l} \frac{\binom{(k-1)/2}{(l-1)/2}^2}{\binom{k-1}{l-1}^2} \frac{4}{2^{2k}} \binom{k-1}{(k-1)/2}^2 = \frac{k}{l} \frac{\binom{(k-1)/2}{(l-1)/2}^2}{\binom{k-1}{l-1}^2} \frac{4}{2^{2k}} \binom{k-1}{(k-1)/2}^2$$

and using $\frac{4}{2^{2k}} \binom{k-1}{(k-1)/2}^2 = \Theta(1/k)$, we obtain

$$W_l(MAJ_k) = \Theta\left(\frac{\binom{(k-1)/2}{(l-1)/2}^2}{k \binom{k-1}{l-1}}\right)$$

for $l \geq (k+1)/2$. In the case $l = (k+1)/2$, we have $W_l(MAJ_k) = \Theta(k^{-3/2})$ using $\binom{a}{a/2} = \Theta(2^a/\sqrt{a})$ for any a . By Stirling's formula, $\frac{\binom{(k-1)/2}{(l-1)/2}^2}{\binom{k-1}{l-1}} \approx \sqrt{\frac{2(k-1)}{\pi(l-1)(k-l)}}$, which is nondecreasing when $l \geq (k+1)/2$, so $W_l(MAJ_k) = \Omega(k^{-3/2})$ for $l \geq (k+1)/2$. ◀

Next, we consider the EXACT-HALF function, $g(x) = 1 \Leftrightarrow |x| = k/2$.

► **Lemma 23.** *Let k be even. Then $W_{\geq k/2}(EXACT-HALF_k) = \Theta(1/\sqrt{k})$.*

Proof. Let $g : \{0,1\}^k \rightarrow \{0,1\}$ be the EXACT-HALF function. It will be convenient for the proof to switch to the representation of the Fourier transform of g that $\widehat{g}(s) = \frac{1}{2^k} \sum_{x \in \{0,1\}^k} (-1)^{s \cdot x} g(x)$, which is equivalent to the representation used in the rest of this paper for all s such that $s \neq 0^k$, up to a constant factor. Then, for $s \neq 0^k$,

$$\widehat{g}(s) = \sum_{x, |x|=k/2} (-1)^{x \cdot s} = \frac{1}{2^k} \sum_{i=0}^{k/2} (-1)^i \binom{|s|}{i} \binom{k-|s|}{i},$$

where the last expression is a Krawtchouk polynomial [33]. This is symmetric about $|s| = k/2$, so

$$\sum_{s, |s| \geq k/2} \widehat{g}(s)^2 \geq \frac{1}{2} \sum_s \widehat{g}(s)^2 = \frac{1}{2} \|g\|_2^2 = \Theta(1/\sqrt{k}).$$

So, by the above lemmas, we reproduce the $\Theta(k^{1/4})$ complexity of Belovs' algorithms for the majority and EXACT-HALF functions up to a logarithmic factor. The algorithms are also time-efficient.

► **Theorem 24.** *There exist quantum algorithms that learn the majority and exact-half functions on k -bits using $O(k^{1/4} \log k)$ queries. The time complexity is $O(nk^{1/4} \log k)$.*

1:22 Quantum Algorithms for Learning a Hidden Graph

Finally, we observe a simple general approach which can be used to solve the learning problem for almost all functions efficiently. Define the influence of the j 'th variable as

$$\text{Inf}_j(g) = \sum_{T \ni j} \widehat{g}(T)^2 = \Pr_{x \in \{0,1\}^k} [g(x) \neq g(x^j)],$$

where x^j is the bit-string equal to x with its j 'th bit flipped.

► **Proposition 25** (essentially Atıcı and Servedio [11]). *Assume that, for all $j \in S$, $\text{Inf}_j(g) \geq \epsilon$. Then there is a quantum algorithm which identifies S with probability $1 - \delta$ using $O(\epsilon^{-1} \log(k/\delta))$ queries to f .*

Proof. We apply Fourier sampling to f , which returns a subset $T \subseteq [k]$ with probability $\widehat{g}(T)^2$. We use this subroutine q times and output the union of the subsets of variables returned. The probability that the j 'th variable is included in each sample is $\text{Inf}_j(g) \geq \epsilon$. The probability that there exists a variable that is not returned after the q queries is at most $k(1 - \epsilon)^q \leq ke^{-q\epsilon}$. So it is sufficient to take $q = O(\epsilon^{-1} \log(k/\delta))$ to learn all the variables except with probability δ . ◀

If g is picked at random, then for all j , $\text{Inf}_j(g)$ is lower-bounded by a constant with high probability. So, by Proposition 25, for almost all functions g , there is a quantum algorithm that identifies S using $O(\log k)$ queries and succeeds with probability 0.99. This holds even if g is unknown, and is an exponential improvement over the optimal classical complexity.

Quantum Algorithm for Stochastic Optimal Stopping Problems with Applications in Finance

João F. Doriguello ✉ 

Centre for Quantum Technologies, National University of Singapore, Singapore

Alessandro Luongo ✉

Centre for Quantum Technologies, National University of Singapore, Singapore

Jinge Bao ✉

Centre for Quantum Technologies, National University of Singapore, Singapore

Patrick Rebenrost ✉

Centre for Quantum Technologies, National University of Singapore, Singapore

Miklos Santha ✉

Centre for Quantum Technologies, National University of Singapore, Singapore

Abstract

The famous least squares Monte Carlo (LSM) algorithm combines linear least square regression with Monte Carlo simulation to approximately solve problems in stochastic optimal stopping theory. In this work, we propose a quantum LSM based on quantum access to a stochastic process, on quantum circuits for computing the optimal stopping times, and on quantum techniques for Monte Carlo. For this algorithm, we elucidate the intricate interplay of function approximation and quantum algorithms for Monte Carlo. Our algorithm achieves a nearly *quadratic* speedup in the runtime compared to the LSM algorithm under some mild assumptions. Specifically, our quantum algorithm can be applied to American option pricing and we analyze a case study for the common situation of Brownian motion and geometric Brownian motion processes.

2012 ACM Subject Classification Mathematics of computing → Stochastic processes; Mathematics of computing → Markov-chain Monte Carlo methods; Theory of computation → Quantum computation theory

Keywords and phrases Quantum computation complexity, optimal stopping time, stochastic processes, American options, quantum finance

Digital Object Identifier 10.4230/LIPIcs.TQC.2022.2

Related Version *Full Version:* <https://arxiv.org/abs/2111.15332> [28]

Funding Research at CQT is funded by the National Research Foundation, the Prime Minister’s Office, and the Ministry of Education, Singapore under the Research Centres of Excellence programme’s research grant R-710-000-012-135.

Acknowledgements We thank Rajagopal Raman for pointing out Ref. [53] and the importance of the Longstaff-Schwarz algorithm in the insurance industry. We also thank Koichi Miyamoto for Ref. [50] and Eric Ghysels for Refs. [15, 16].

1 Introduction

Within stochastic optimization, optimal stopping theory is a broad area of applied mathematics that started in the 1940s and 1950s mainly with A. Wald [90] and is concerned with the problem of deciding the best time to “stop” or take an action in order to maximize an expected reward [81]. The time at which the observations are terminated, called *stopping time*, is a random variable depending on the data observed so far in the process. A simple



© João F. Doriguello, Alessandro Luongo, Jinge Bao, Patrick Rebenrost, and Miklos Santha; licensed under Creative Commons License CC-BY 4.0

17th Conference on the Theory of Quantum Computation, Communication and Cryptography (TQC 2022).

Editors: François Le Gall and Tomoyuki Morimae; Article No. 2; pp. 2:1–2:24

Leibniz International Proceedings in Informatics



LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

example of an optimal stopping problem is the following: consider a game in which 100 numbers are written on 100 pieces of paper without restrictions on the numbers, except that no number appears more than once. The pieces of paper are shuffled faced down and you are asked to look at the numbers, without having seen them before and *once* at a time, and to stop when you think that you have found the biggest number. It turns out that there is a stopping rule that allows you to stop at the biggest number for $1/e$ fraction of the inputs.

Since its conception, optimal stopping theory collected problems from many disparate areas under a unique umbrella [71], e.g. quickest detection [82], sequential parameter estimation [64] and sequential hypothesis testing [26]. Probably the most famous optimal stopping problem is the one of option pricing in finance, especially American options [57]. A central problem in the world of finance is to assign a monetary value to a hitherto unvalued asset. In the capital markets, there exists a large variety of financial assets which are derivative to underlying assets such as stocks, bonds, or commodities. One of the most well-known examples is the European call option which allows the buyer to “lock in” a price for buying a stock at some future time (or “exercise” time). A fair valuation of such an option was first discussed in the seminal works of Black and Scholes [11] and Merton [61]. In the times since, the methods proposed in these works have become standard practice in the financial sector and have been extended and generalized for many financial derivatives and market models.

American options allow the buyer to exercise the option at any point in time between the time of purchase and a fixed final time. In contrast to an European option, there are no known closed formulas for the price of an American option with finite maturity date even in simple models like the Black-Scholes-Merton one. Theoretically, an American option can be viewed as a stochastic optimal stopping problem for the buyer and a super-martingale hedging problem for the seller [31]. Practical algorithms have been developed for the pricing of American options [23, 77, 48, 46], an important class being least squares Monte Carlo (LSM) algorithms originally proposed independently by Tsitsiklis and Van Roy [87] and by Longstaff and Schwartz [57].

Among the aforementioned classical algorithms for option pricing – and other topics in finance – is the sub-field of quantum computing of designing quantum algorithms in the context of financial problems [67, 12, 29], e.g. risk management [92], financial greeks [84, 3], portfolio optimization [6, 25, 45, 75, 42, 2] and option pricing [59, 89]. A common tool in obtaining a quantum advantage is amplitude estimation [14] and its generalizations for Monte Carlo sampling [63, 41, 22, 21]. A few different works devised quantum algorithms for derivative pricing based on quantum subroutines for Monte Carlo [74, 83, 18], e.g. European [73, 32, 72] and American/Bermudan [62] option pricing, and option pricing in the local volatility model [49, 3] (of which the Black-Scholes model is a subcase). Given its versatility and previous cases of success, it is only natural to explore the applicability of quantum methods for Monte Carlo to problems in optimal stopping theory. In this work we focus on tailoring these methods to LSM algorithms.

Applications of LSM algorithm

Among the whole domain of optimal stopping problems, there are many that can be approached directly with LSM, e.g. the secretary problem [24], modelling the optimal time to call an election based on data [86], estimating the solvency of governments with respect to their debt [79], and multi-armed bandit problems [40]. Another important application of LSM is in the insurance sector. In fact, LSM can be used to estimate the VaR (Value at Risk) [53] and life insurance contracts [5] (see also [69] for a comparison of LSM with other methods). The computational challenges of this domain were further highlighted by recent European

regulatory requirements [1, 27]. LSM is also often used for solving Backward Stochastic Differential Equations (BSDE). Some numerical algorithms for BSDE are two-steps stochastic procedures involving a discretisation step where the solutions obtained at time t of the BSDE are projected onto a space obtained from the filtrations at time $t - 1$. This step involves a conditional expectation that cannot be calculated analytically, but must be estimated using some approximation procedure. The idea of applying LSM to BSDE was first introduced in [35] and further developed in [36, 54]. Recently, this method has been generalized to solve two-dimensional forward-backward stochastic differential equations [55, 8].

Other algorithms for option pricing

LSM is not the only type of algorithm that can be used to price American options [30]. Besides a few attempts to give an analytical formula under certain conditions [51], the vast majority of them has been directed towards giving numerical results, which we briefly discuss in this section. A simple and well-known way of pricing American options is through the use of binomial trees. While the origins of this technique are somewhat unclear [19], the first articles that proposed the idea of binomial trees for pricing options are considered to be [23, 77], with the first seminal ideas proposed in the first edition of [80]. McKean [60] realized that the price of an American option can be cast as a free boundary problem [70], which is a particular partial differential equation that can be solved numerically. There is a flurry of other methods to price American options based on partial differential equations. We name a few approaches such as variational inequalities [48, 9], linear complementary [46], and those related to free boundaries [88]. However, as noted in [30], these methods often suffer from the curse of dimensionality, as they require the computing time and the storage to grow exponentially with the dimension of the underlying state space. LSM is also not the only Monte Carlo approach for pricing American options. One of the first works using Monte Carlo for option pricing is [13]. Reviews of Monte Carlo and other methods for the problem of American option pricing can be found in [33, 52, 17]. In contrast to giving lower bounds for the true optimal stopping value – as the LSM algorithm – Rogers [78] proposed a method which leverages a dual problem, resulting in an upper bound for the optimal stopping value. Last but not least, semi-analytical approaches for American option pricing and optimal stopping time are also used [7].

1.1 Problem statement

Optimal stopping theory is concerned with the problem of finding the best moment to stop a process in order to maximize an expected reward. More generally, assume a discrete-time stochastic process $\mathbf{X} = (X_t)_{t=0}^T$ (which corresponds to the market model in financial applications), assumed to be Markovian, defined on a filtered probability space $(\Omega, \mathcal{F}, (\mathcal{F}_t)_{t=0}^T, \mathbb{P})$ and with state space (E, \mathcal{E}) , where $E \subseteq \mathbb{R}^d$. We shall assume that \mathbf{X} is *adapted* with respect to $(\mathcal{F}_t)_{t=0}^T$, meaning that each X_t is \mathcal{F}_t -measurable, and that $X_0 = x_0$ is deterministic, therefore, sometimes we write the Markov chain $(X_t)_{t=1}^T$ as starting from $t = 1$. Each element X_t for $t \in \{0, \dots, T\}$, called the underlying process at time t , gives rise to an image probability measure (also called pushforward measure) ρ_t in $E \subseteq \mathbb{R}^d$, i.e., $\rho_t(Y) = \mathbb{P}[\omega \in \Omega : X_t(\omega) \in Y]$ for any $Y \in \mathcal{E}$ (note that ρ_0 is the probability measure that assigns measure 1 to the singleton set containing x_0). We denote by $\mathbf{X}_t = (X_j)_{j=t}^T$ the last $T - t + 1$ random variables in the stochastic process. Let $L^2(E, \rho_t) = L^2(\rho_t)$ be the set of squared integrable functions with norm $\|f\|_{L^2(\rho_t)} := \sqrt{\mathbb{E}_{\rho_t}[|f(X_t)|^2]}$ and define the uniform norm $\|f\|_u = \sup\{|f(s)| : s \in E\}$ for $f : E \rightarrow \mathbb{R}$. Consider further a *payoff process*: a non-negative adapted process $(Z_t)_{t=0}^T$

on the filtered probability space $(\Omega, \mathcal{F}, (\mathcal{F}_t)_{t=0}^T, \mathbb{P})$ obtained from $(X_t)_{t=0}^T$ as $Z_t = z_t(X_t)$ for square-integrable real functions $z_t \in L^2(E, \rho_t)$, for $t \in \{0, \dots, T\}$. Moreover, given an event \mathcal{E} , we denote by $\mathbf{1}\{\mathcal{E}\}$ the indicator function $\mathbf{1}\{\mathcal{E}\} = 1$ if \mathcal{E} is true and 0 if not. Finally, we shall use \mathbb{P} to denote the probability measure behind the stochastic process $(X_t)_{t=0}^T$, and the probability notation \Pr for other processes, e.g. the outcome probability of a quantum measurement.

Our main problem is when to stop the process and take the payoff so that the expected payoff is maximized. This is formalized by the idea of *stopping time*, which is a random variable that selects one of the possible times $\{0, 1, \dots, T\} \cup \{+\infty\}$ and satisfies a measurability condition.

► **Definition 1** (Stopping time). *A stopping time is a function $\tau : \Omega \rightarrow \{0, 1, \dots, T\} \cup \{+\infty\}$ such that $\{\omega \in \Omega \mid \tau(\omega) = t\} \in \mathcal{F}_t$ for $t \in \{0, \dots, T\}$. The payoff obtained by using τ is $Z_\tau(\omega) := Z_{\tau(\omega)}(\omega)$. Let $\mathbb{T}_t := \{\tau \mid \tau \text{ is a stopping time with } t \leq \tau \leq T\}$ be the set of all stopping times taking values in $[t, T]$. A stopping time $\tau^* \in \mathbb{T}_t$ is called optimal in the interval $[t, T]$ if*

$$\mathbb{E}[Z_{\tau^*} | X_t] = \operatorname{ess\,sup}_{\tau \in \mathbb{T}_t} \mathbb{E}[Z_\tau | X_t].$$

The maximization is expressed via an essential supremum such that the null sets of the probability measure do not affect the result. For more details on the essential supremum see [31, Appendix A.5]. The optimal stopping problem then consists in finding a stopping time τ that maximizes the *expected* value payoff.

► **Problem 2** (Optimal stopping problem). *Let $(Z_t)_{t=0}^T$ be a payoff process. For $\epsilon_{\text{final}} > 0$, approximate the exact value $\sup_{\tau \in \mathbb{T}_0} \mathbb{E}[Z_\tau]$ to additive accuracy ϵ_{final} with high probability.*

A well-studied solution strategy for the above problem statement is based on dynamic programming for a set of stopping times. A crucial concept is the *Snell envelope* [65, 31] $\mathcal{U}_t : \Omega \rightarrow \mathbb{R}$ of a payoff process $Z_t = z_t(X_t)$ (for some $z_t \in L^2(E, \rho_t)$, $t \in \{0, \dots, T\}$) defined as

$$\begin{cases} \mathcal{U}_T = Z_T, \\ \mathcal{U}_t = \max \{Z_t, \mathbb{E}[\mathcal{U}_{t+1} | X_t]\}, \quad 0 \leq t \leq T-1. \end{cases} \quad (1)$$

Define the stopping times $\tau_t := \min\{u \geq t \mid \mathcal{U}_u = Z_u\}$. The Snell envelope is related to the maximal expected payoff according to the next theorem: τ_t maximizes the expectation of Z_τ among all $\tau \in \mathbb{T}_t$, i.e., that τ_t are optimal stopping times (in their respective intervals).

► **Theorem 3** ([31, Theorem 6.18]). *The Snell envelope \mathcal{U}_t of Z_t satisfies*

$$\mathcal{U}_t = \mathbb{E}[Z_{\tau_t} | X_t] = \operatorname{ess\,sup}_{\tau \in \mathbb{T}_t} \mathbb{E}[Z_\tau | X_t].$$

In particular, $\mathcal{U}_0 = \mathbb{E}[Z_{\tau_0}] = \sup_{\tau \in \mathbb{T}_0} \mathbb{E}[Z_\tau] = \max\{Z_0, \mathbb{E}[Z_{\tau_1}]\}$.

Hence, finding an approximate \mathcal{U}_0 solves our Problem 2. In order to solve the dynamic programming behind the Snell envelope, it is more convenient to recast the dynamic programming in terms of the optimal stopping times τ_t (rather than in terms of value functions) as follows.

► **Theorem 4.** *The dynamic programming principle in Eq. (1) can be recast in terms of the stopping times $\tau_t = \min\{u \geq t \mid \mathcal{U}_u = Z_u\}$ as*

$$\begin{cases} \tau_T = T, \\ \tau_t = t \mathbf{1}\{Z_t \geq \mathbb{E}[Z_{\tau_{t+1}} | X_t]\} + \tau_{t+1} \mathbf{1}\{Z_t < \mathbb{E}[Z_{\tau_{t+1}} | X_t]\}, \quad 0 \leq t \leq T-1. \end{cases}$$

Proof. The case $t = T$ is trivial. Assume $t < T$. Note that $\mathbb{E}[Z_{\tau_{t+1}}|X_t] = \mathbb{E}[\mathbb{E}[Z_{\tau_{t+1}}|X_{t+1}]|X_t]$ because of the tower property of the expectation value with the filtration generated by X_t . In addition, $\mathbb{E}[Z_{\tau_{t+1}}|X_{t+1}] = \mathcal{U}_{t+1}$ from Theorem 3. Hence, if $Z_t \geq \mathbb{E}[Z_{\tau_{t+1}}|X_t]$, then $Z_t \geq \mathbb{E}[\mathcal{U}_{t+1}|X_t]$. This latter statement, by the definition of the Snell envelope, implies $\mathcal{U}_t = Z_t$ and then $\tau_t = t$. On the other hand, if $Z_t < \mathbb{E}[Z_{\tau_{t+1}}|X_t]$, then, $Z_t < \mathbb{E}[\mathcal{U}_{t+1}|X_t] \implies Z_t \neq \mathcal{U}_t$, and so $\tau_t = \min\{u \geq t \mid Z_u = \mathcal{U}_u\} = \min\{u \geq t + 1 \mid Z_u = \mathcal{U}_u\} = \tau_{t+1}$. \blacktriangleleft

The stopping time τ_0 thus maximizes $\mathbb{E}[Z_\tau]$ in Problem 2. The quantities $\mathbb{E}[Z_{\tau_{t+1}}|X_t]$ are called *continuation values*. In the past, many different approaches were developed to tackle the dynamic programming above [51, 19, 23, 77, 80, 60, 70, 48, 9, 46, 88, 13, 33, 52, 17]. A famous approach is the least squares Monte Carlo (LSM) by Longstaff and Schwartz [57].

2 The least squares Monte Carlo algorithm

The LSM algorithm consists in solving the dynamic programming in Theorem 4 by means of two approximations. The first one is to approximate the continuation values $\mathbb{E}[Z_{\tau_{t+1}}|X_t]$ using a set of measurable real-valued functions in $L^2(E, \rho_t)$, e.g. by projection onto a finite-dimensional set of linearly independent polynomials. Let $\mathcal{H}_0 \subseteq \mathbb{R}$ and let, for $t \in [T - 1]$, $\mathcal{H}_t \subseteq L^2(E, \rho_t)$ be a subset of real-valued functions on E , called *approximation architecture* or hypothesis class, that will be used to approximate the continuation values. By approximating $\mathbb{E}[Z_{\tau_{t+1}}|X_t]$ by $f_t \in \mathcal{H}_t$ for each $t \in \{0, \dots, T - 1\}$, we can write the approximate dynamic programming as

$$\begin{cases} \tilde{\tau}_T = T, \\ \tilde{\tau}_t = t\mathbf{1}\{Z_t \geq f_t\} + \tilde{\tau}_{t+1}\mathbf{1}\{Z_t < f_t\}, \quad 0 \leq t \leq T - 1. \end{cases} \quad (2)$$

Note that $\tilde{\tau}_t = \tilde{\tau}_t(f_t, \dots, f_{T-1})$ depends on the approximation architecture.

The second approximation of the algorithm is to numerically evaluate the approximations f_t in $L^2(\rho_t)$ by a Monte Carlo procedure. We sample N independent paths $(X_t^{(1)})_{t=0}^T, \dots, (X_t^{(N)})_{t=0}^T$ of the Markov chain $\mathbf{X} = (X_t)_{t=0}^T$ and denote by $Z_t^{(n)} = z_t(X_t^{(n)})$ the associated payoffs conditioned on $X_t^{(n)}$, where $z_t \in L^2(E, \rho_t)$, $t \in \{0, \dots, T\}$. Write the random variables of the last $T - t + 1$ elements of all the sampled Markov chains by $\mathbf{X}_t^{(N)} = (X_t^{(1)}, \dots, X_t^{(N)}, X_{t+1}^{(1)}, \dots, X_{t+1}^{(N)}, \dots, X_T^{(1)}, \dots, X_T^{(N)})$. For each path, the dynamic programming in Eq. (2) is solved recursively by approximating the continuation values in \mathcal{H}_t via a least square estimator. The result is sampled stopping times $\tilde{\tau}_t^{(n)}$ that $\tilde{\tau}_t$ takes on each random path. We stress that, due to the recursive nature of Eq. (2), the stopping times $\tilde{\tau}_t^{(n)}$ will depend on $\mathbf{X}_t^{(N)}$, and consequently also the payoffs $Z_{\tilde{\tau}_t^{(n)}}^{(n)} = z_{\tilde{\tau}_t^{(n)}}(\mathbf{X}_t^{(N)})(X_{\tilde{\tau}_t^{(n)}}^{(n)}(\mathbf{X}_t^{(N)}))$.

The dependence on $\mathbf{X}_t^{(N)}$ should be clear from the context and therefore we shall simply write $Z_{\tilde{\tau}_t^{(n)}}^{(n)}$. In summary, combining both the approximation architecture and the Monte Carlo sampling, at each $t \in [T - 1]$ we take $f_t \in \mathcal{H}_t$, depending on $\mathbf{X}_t^{(N)}$, satisfying

$$\frac{1}{N} \sum_{n=1}^N \left(Z_{\tilde{\tau}_{t+1}^{(n)}}^{(n)} - f_t(X_t^{(n)}) \right)^2 \leq \epsilon + \inf_{g \in \mathcal{H}_t} \frac{1}{N} \sum_{n=1}^N \left(Z_{\tilde{\tau}_{t+1}^{(n)}}^{(n)} - g(X_t^{(n)}) \right)^2 \quad (3)$$

for some given $\epsilon \geq 0$. It might be the case that an exact minimizer of the above optimization problem does not exist (the infimum does not belong to \mathcal{H}_t) or is hard to compute, meaning that an ϵ -approximation could be used. Given the choice of $f_t \in \mathcal{H}_t$, it is then used

in Eq. (2) to obtain $\tilde{\tau}_t^{(n)}$, and so on recursively. At the end of the recursion we can take $f_0 = \frac{1}{N} \sum_{n=1}^N Z_{\tilde{\tau}_1^{(n)}}^{(n)}$ as an exact minimizer, since X_0 is constant, and obtain the approximation $\tilde{\mathcal{U}}_0$ to \mathcal{U}_0 as

$$\tilde{\mathcal{U}}_0 = \max \left\{ Z_0, \frac{1}{N} \sum_{n=1}^N Z_{\tilde{\tau}_1^{(n)}}^{(n)} \right\}. \quad (4)$$

In this paper we shall be particularly interested in finite-dimensional linear approximation architectures, for which an exact minimizer exists in Eq. (3). Consider then a set $\{e_{t,k} : E \rightarrow \mathbb{R}\}_{k=1}^m$ of m linearly independent measurable real functions and take the vector space generated by them as our approximation architecture \mathcal{H}_t , $t \in [T-1]$. Therefore, the infimum in Eq. (3) is attained by projecting the continuation values onto \mathcal{H}_t as $\alpha_t \cdot \vec{e}_t(X_t)$, where $\vec{e}_t(\cdot) := (e_{t,1}(\cdot), \dots, e_{t,m}(\cdot))^\top$ and the m -dimensional vector α_t , the projection coefficients, is the least square estimator given by [20]

$$\alpha_t = \arg \min_{a \in \mathbb{R}^m} \mathbb{E}[(Z_{\tilde{\tau}_{t+1}} - a \cdot \vec{e}_t(X_t))^2].$$

Given the assumption that $\{e_{t,k}\}_{k=1}^m$ are linearly independent for each $t \in [T-1]$, the vector $\alpha_t \in \mathbb{R}^m$ has the explicit expression

$$\alpha_t = A_t^{-1} b_t \text{ where } b_t = \mathbb{E}[Z_{\tilde{\tau}_{t+1}} \vec{e}_t(X_t)] \quad (5)$$

and the $m \times m$ matrix A_t has coefficients

$$(A_t)_{k,l} = \mathbb{E}[e_{t,k}(X_t)e_{t,l}(X_t)]. \quad (6)$$

Often it is hard to compute α_t and A_t exactly. As previously mentioned, the LSM algorithm approximates these by Monte Carlo sampling,

$$\tilde{\alpha}_t = \tilde{A}_t^{-1} \frac{1}{N} \sum_{n=1}^N Z_{\tilde{\tau}_{t+1}^{(n)}} \vec{e}_t(X_t^{(n)}) \quad (7)$$

and

$$(\tilde{A}_t)_{k,l} = \frac{1}{N} \sum_{n=1}^N e_{t,k}(X_t^{(n)})e_{t,l}(X_t^{(n)}). \quad (8)$$

More generally, though, any good approximation $\tilde{\alpha}_t$ and \tilde{A}_t to α_t and A_t , respectively, is valid, and we shall not restrict the notation $\tilde{\alpha}_t$ and \tilde{A}_t to only mean the above sampled quantities.

We have introduced the quantities that are important for the LSM algorithm. To present the algorithm, we first specify the input model.

► **Definition 5** (Sampling access to Markov chain). *Given a Markov chain $(X_t)_{t=1}^T$ on a probability space (Ω, \mathbb{P}) and with state space $E \subseteq \mathbb{R}^d$, we define sampling access as the ability to draw a sample $\omega \in \Omega$ according to \mathbb{P} and observe the value $X_t(\omega)$ for some $t \in [T]$. One sample costs time $\mathcal{T}_{\text{samp}}$.*

► **Definition 6** (Query access to function). *Let $E \subseteq \mathbb{R}^d$ and $h : E \rightarrow \mathbb{R}$ be a function. We define query access as the ability to observe the value $h(x)$ for any given $x \in E$. One query costs time \mathcal{T}_h .*

■ **Algorithm 1** Classical LSM algorithm for optimal stopping problem.

Input: Integer $N \in \mathbb{N}$. Sampling access to Markov chain $(X_t)_{t=0}^T$ defined on a sample space Ω and with state space $E \subseteq \mathbb{R}^d$. Query access to functions $\{z_t : E \rightarrow \mathbb{R}\}_{t=0}^T$ and $\{e_{t,k} : E \rightarrow \mathbb{R}\}_{t \in [T-1], k \in [m]}$, where $\{e_{t,k}\}_{k=1}^m$ are linearly independent for each $t \in [T-1]$. Let $\vec{e}_t(\cdot) := (e_{t,1}(\cdot), \dots, e_{t,m}(\cdot))^\top$.

- 1: Sample N independent paths $(X_t^{(1)}, \dots, X_t^{(N)})_{t=0}^T$.
- 2: Query payoffs $(Z_t^{(1)}, \dots, Z_t^{(N)})_{t=0}^T$ and values $(e_{t,k}(X_t^{(1)}), \dots, e_{t,k}(X_t^{(N)}))_{t \in [T-1], k \in [m]}$.
- 3: Compute the matrices $\{\tilde{A}_t\}_{t=1}^{T-1}$ with entries as in Eq. (8).
- 4: Compute the inverses $\{\tilde{A}_t^{-1}\}_{t=1}^{T-1}$.
- 5: Set $\tilde{\tau}_T^{(n)} = T$ for $n \in [N]$.
- 6: **for** $t = T - 1$ to 1 **do**
- 7: Calculate the vector $\tilde{\alpha}_t = \tilde{A}_t^{-1} \frac{1}{N} \sum_{n=1}^N Z_{\tilde{\tau}_{t+1}^{(n)}}^{(n)} \vec{e}_t(X_t^{(n)})$.
- 8: Calculate $\tilde{\tau}_t^{(n)} = t \mathbf{1}\{Z_t^{(n)} \geq \tilde{\alpha}_t \cdot \vec{e}_t(X_t^{(n)})\} + \tilde{\tau}_{t+1}^{(n)} \mathbf{1}\{Z_t^{(n)} < \tilde{\alpha}_t \cdot \vec{e}_t(X_t^{(n)})\}$, $n \in [N]$.
- 9: **end for**
- 10: Output $\tilde{U}_0 := \max \left\{ Z_0, \frac{1}{N} \sum_{n=1}^N Z_{\tilde{\tau}_1^{(n)}}^{(n)} \right\}$.

Here, we assume that the functions of the approximation architecture and functions for the payoff take time \mathcal{T}_e and \mathcal{T}_z , respectively, to evaluate. Both sampling and function access have natural quantum extensions, as will be defined in Section 3.

We are now in the position to present the classical LSM algorithm in Algorithm 1. Since we focus on the case where the approximation architectures \mathcal{H}_t are finite-dimensional and linear, we write Algorithm 1 for this particular case.

3 Quantum least squares Monte Carlo algorithm

In this section we shall present our quantum algorithm, which is based on the classical LSM algorithm (Algorithm 1). Before we discuss it, we review our computational model, input assumptions, and the quantum algorithm for Monte Carlo used in this work. In what follows, for simplicity, we suppose that $|0\rangle$ describes a register with sufficiently many qubits initialized in the all-0 state.

3.1 Computational model

In this subsection, we address our quantum computational model. We work in the standard circuit model of quantum computation [66]. Aside from these standard assumptions, we take the following additional assumptions on the computational model.

Arithmetic model

In our work, we perform the arithmetic computations on the quantum computer by using a fixed point representation for real numbers. We assume that we can have enough qubits for storing these numbers, represented as bit strings using the following definition. We also assume to work with enough precision so that numerical errors in the computation are negligible, and will not impact the final output of our algorithm.

► **Definition 7** (Fixed-point encoding of real numbers [76]). *Let c_1, c_2 be positive integers, and $a \in \{0, 1\}^{c_1}$, $b \in \{0, 1\}^{c_2}$, and $s \in \{0, 1\}$ be bit strings. Define the rational number as:*

$$\mathcal{Q}(a, b, s) := (-1)^s \left(2^{c_1-1} a_{c_1} + \cdots + 2a_2 + a_1 + \frac{1}{2}b_1 + \cdots + \frac{1}{2^{c_2}}b_{c_2} \right) \in [-R, R], \quad (9)$$

where $R := 2^{c_1} - 2^{-c_2}$. If c_1, c_2 are clear from the context, we can use the shorthand notation for a number $z := (a, b, s)$ and write $\mathcal{Q}(z)$ instead of $\mathcal{Q}(a, b, s)$. Given an n -dimensional vector $v \in (\{0, 1\}^{c_1} \times \{0, 1\}^{c_2} \times \{0, 1\})^n$ the notation $\mathcal{Q}(v)$ means an n -dimensional vector whose j -th component is $\mathcal{Q}(v_j)$, for $j \in [n]$.

The choice of values for c_1 and c_2 depends on the choice of input functions used when running the algorithm. For the purposes of optimizing the quantum circuit, these constants can be changed dynamically in various steps of the computation. While analyzing how error propagates and accumulates throughout the operations in the quantum circuit is essential to ensure a correct estimation of the final result, this analysis can only be done for a given choice of input functions. We avoid the analysis of such details by using the quantum arithmetic model as in Definition 8. A standard result is that any Boolean function can be reversibly computed. Any reversible computation can be realized with a circuit involving negation and three-bit Toffoli gates. Such a circuit can be turned into a quantum circuit with single-qubit NOT gates and three-qubit Toffoli gates. Since most circuits for arithmetic operations operate with a number of gates of $O(\text{poly}(c_1, c_2))$ this implies a number of quantum gates of $O(\text{poly}(c_1, c_2))$ for the corresponding quantum circuit.

► **Definition 8** (Quantum arithmetic model). *Given $c_1, c_2 \in \mathbb{N}$ specifying fixed-point precision numbers as in Definition 7, we say we use a quantum arithmetic model of computation if the four arithmetic operations can be performed in constant time in a quantum computer.*

In our computational model we do not include the cost for performing operations described in our arithmetic model. For instance, a central computational step of the quantum algorithm is the circuit computing the stopping times $\tilde{\tau}_t(x)$, but as the circuit depth depends polynomially on c_1 and c_2 , we do not take into account this cost when stating our runtime. For an example of a resource estimation for a financial problem that takes into account the cost of arithmetic operations in fixed-point precision, we refer to [18].

Quantum input access

We assume that we have quantum oracles for certain input functions. The classical algorithm assumes access to two different kinds of oracles. The first is an oracle that allows us to obtain samples from the Markov chain $(X_t)_{t=0}^T$. The second kind of oracle is evaluating the functions $\{z_t\}_{t=0}^T$ and $\{e_{t,k}\}_{t \in [T-1], k \in [m]}$. We assume access to the quantum versions of these oracles (formalized below). The first kind of quantum oracle prepares a quantum state that is in a superposition over the different outcomes of the Markov chain, weighted by amplitudes which are square roots of their classical probabilities. A measurement in the computational basis of such a state obtains a single sample with the corresponding probability and hence directly recovers a single use of the classical sampling access. The second kind of quantum oracle evaluates a given function in superposition over its inputs. While the functions $\{e_{t,k}\}_{t \in [T-1], k \in [m]}$ are usually chosen to be low-degree polynomials (and thus admit efficient classical and quantum circuits with gate complexity proportional to the degree of the polynomial), the functions $\{z_t\}_{t=0}^T$ might be arbitrarily complex. Usually the complexity of these functions is not discussed in classical literature, and we use placeholders for their evaluation cost.

► **Definition 9** (Quantum sampling access to a Markov chain). *Let $(X_t)_{t=1}^T$ be a Markov chain defined on a filtered probability space $(\Omega, \mathcal{F}, (\mathcal{F}_t)_{t=1}^T, \mathbb{P})$, for a finite Ω , assuming values in a finite state space $E \subseteq \mathbb{R}^d$. Given $x \in E^T$, let $p(x) := \mathbb{P}[X_1 = x_1] \prod_{t=1}^{T-1} \mathbb{P}[X_{t+1} = x_{t+1} | X_t = x_t]$ be the probability that $X_1 = x_1, \dots, X_T = x_T$. Let \mathcal{H} be a finite-dimensional Hilbert space with basis $\{|x\rangle\}_{x \in E^T}$. We say that we have quantum sampling access to $(X_t)_{t=1}^T$ if we are given an oracle U on \mathcal{H} such that $U|0\rangle = \sum_{x \in E^T} \sqrt{p(x)}|x\rangle$. One application of U costs $T \cdot \mathcal{T}_{\text{samp}}$ time. If $T = 1$, we say that we have quantum sampling access to a random variable X if we are given an oracle U on \mathcal{H} such that $U|0\rangle = \sum_{x \in E} \sqrt{p(x)}|x\rangle$, where $p(x) := \mathbb{P}[X = x]$ is the probability that $X = x$.*

We note the alternative definition of the unitary U such that $U|0\rangle = \sum_{x \in E^T} \sqrt{p(x)}|x\rangle|\psi_x\rangle$ for unknown garbage unit states $|\psi_x\rangle$. Such garbage states do not change our analysis, so we shall ignore them and work with the unitary U from Definition 9.

Even though we assume the existence of the oracle U , constructing such unitary is an important question on its own. A few methods have been proposed in order to tackle such problem, one of the most famous is due to Grover and Rudolph [39] (see [58] for recent improvements on the Grover-Rudolph method), which loads into a quantum computer a discretization of a distribution with density function $p(x)$. More specifically, it creates the quantum state $\sum_{i=0}^{2^n-1} \sqrt{p_i^{(n)}}|i\rangle$ with $p_i^{(n)} = \int_{x_i^{(n)}}^{x_{i+1}^{(n)}} p(x)dx$ by recursively (on n) computing quantities like $f_n(i) = \int_{x_i^{(n)}}^{(x_i^{(n)}+x_{i+1}^{(n)})/2} p(x)dx / \int_{x_i^{(n)}}^{x_{i+1}^{(n)}} p(x)dx$ for $i = 0, \dots, 2^n - 1$. It is also possible to perform simple Taylor approximations on $f_n(i)$ when n is sufficiently large (see [49, Equation (35)]). We briefly note that the issues about the Grover-Rudolph method recently pointed out by [44] only arise when one needs to *sample* from the distribution $p(x)$ in order to compute $f_n(i)$, which is not the case in many situations, e.g. in finance.

► **Definition 10** (Quantum access to a function). *Let $E \subseteq \mathbb{R}^d$ be a finite set and let \mathcal{H} be a Hilbert space with basis $\{|x\rangle\}_{x \in E}$. Given $h : E \rightarrow \{0, 1\}^n$, we say that we have (V_h, \mathcal{T}_h) -quantum access to h if we have access to a quantum circuit V_h on $\mathcal{H} \otimes \mathbb{C}^{2^n}$ such that $V_h|x\rangle|b\rangle = |x\rangle|b \oplus h(x)\rangle$ for any bit string $b \in \{0, 1\}^n$. One application of V_h costs time \mathcal{T}_h .*

Access to quantum controlled rotations

Controlled rotations are a central step in the quantum algorithm for Monte Carlo (Theorem 13). The cost of a controlled rotation depends directly on the number of bits used to specify the angle of rotation [91]. In our computational model we assume that controlled rotations come with a unit cost.

► **Definition 11** (Access to quantum controlled rotations). *We say that we have access to quantum controlled rotations if we have a quantum circuit R whose application takes constant time and, for all rational numbers $x \in [0, 1]$ defined by a $(1 + c_2)$ -bit string in our fixed-point arithmetic model 8, operates as:*

$$R|x\rangle|0\rangle = |x\rangle(\sqrt{1-x}|0\rangle + \sqrt{x}|1\rangle). \tag{10}$$

We note that again this definition allows us to neglect terms $O(c_1 + c_2)$ in the runtime and to neglect complications arising from the arithmetic computation of the arcsin. The access to these rotation unitaries leads to the following fact.

► **Fact 12** (Controlled rotations of a function with an interval). *Consider a rational number representation from Definition 7 for some $c_1, c_2 \in \mathbb{N}$. Assume access to controlled rotations according to Definition 11. Assume (V_h, \mathcal{T}_h) -quantum access to a function h according to Definition 10. For any two bit strings $a, b \in \{0, 1\}^n$, with $0 \leq \mathcal{Q}(a) < \mathcal{Q}(b)$, we can construct a unitary operator $R_{a,b}^h$ on $\mathcal{H} \otimes \mathbb{C}^2$, such that, for all $x \in E$,*

$$R_{a,b}^h |x, 0\rangle = \begin{cases} |x\rangle \left(\sqrt{1 - \frac{\mathcal{Q}(h(x))}{\mathcal{Q}(b)}} |0\rangle + \sqrt{\frac{\mathcal{Q}(h(x))}{\mathcal{Q}(b)}} |1\rangle \right) & \text{if } \mathcal{Q}(a) \leq \mathcal{Q}(h(x)) \leq \mathcal{Q}(b), \\ |x, 0\rangle & \text{otherwise,} \end{cases}$$

where an application of $R_{a,b}^h$ costs $O(\mathcal{T}_h)$ time.

Proof. The quantum circuits for the division and for checking the interval run in constant time in the quantum arithmetic model. The quantum circuits allow us to prepare $|x\rangle |\mathcal{Q}(h(x))/\mathcal{Q}(b)\rangle$ on the interval using V_h two times, where the second register is of size polynomial in c_1 and c_2 . Performing a controlled rotation of an ancilla costs constant time by Definition 11. ◀

Quantum algorithm for Monte Carlo

Our quantum algorithm requires the computation of several expectation values. In this work we use the quantum algorithm for Monte Carlo from Montanaro [63], already adapted to our computational model.

► **Theorem 13** (Quantum algorithm for Monte Carlo `QMonteCarlo`, [63, Theorem 2.5]). *Let X be a random variable given via quantum sampling access as in Definition 9. Consider a rational number representation from Definition 7 for some $c_1, c_2 \in \mathbb{N}$. Let $E \subseteq \mathbb{R}^d$ be a finite set and let \mathcal{H} be a Hilbert space with basis $\{|x\rangle\}_{x \in E}$. Consider a function $h : E \rightarrow \{0, 1\}^n$ via quantum access to the controlled rotations as in Fact 12, where $n \in \mathbb{N}$ such that $c_1 + c_2 + 1 = n$, and each access costs time \mathcal{T}_h . Assume that the random variable $\mathcal{Q}(h(X))$ has finite mean μ , and variance upper-bounded by σ^2 for some known $\sigma > 0$. Given $\delta, \epsilon \in (0, 1)$, there is a quantum algorithm, called `QMonteCarlo`($h(X), \epsilon, \delta, \sigma$), that runs in $O((\sigma/\epsilon) \log(1/\delta) \log^{3/2}(\sigma/\epsilon) \log \log(\sigma/\epsilon)) \times (T\mathcal{T}_{\text{samp}} + \mathcal{T}_h)$ time and outputs an estimate $\tilde{\mu}$ such that $\Pr[|\tilde{\mu} - \mu| \geq \epsilon] \leq \delta$.*

The above result will be used to approximate expected values, e.g. $\mathbb{E}[e_{t,k}(X_t)e_{t,l}(X_t)]$ and $\mathbb{E}[Z_{\tilde{\tau}_{t+1}} \tilde{e}_t(X_t)]$, and was chosen for its simplicity. It is possible, though, to use other, more complicated quantum subroutines for Monte Carlo, e.g. [41, 22, 21]. Refs. [22, 21] propose quantum algorithms for multivariate Monte Carlo estimation, which could be particularly suitable in our case, since most of our quantities of interest are vectors and matrices. However, since these more complex and alternative quantum subroutines for Monte Carlo lead to the same time complexities up to polylogarithm factors as Theorem 13, we decided to use the above result.

Quantum circuits for the stopping times

Recall that Theorem 4 allows us to formulate the stochastic optimal stopping problem with dynamic programming for the optimal stopping times. Having introduced the quantum computational model, we are now in the position to construct quantum circuits for various computations related to the dynamic programming. In particular, we construct a unitary

that propagates backwards the optimal stopping time by one time step according to Eq. (2). In what follows, given a path $x \in E^T$, by $z_{\tilde{\tau}_t(x)}$ we mean $z_{\tilde{\tau}_t(x)}(x_{\tilde{\tau}_t(x)})$, i.e., the associated payoff of the $\tilde{\tau}_t(x)$ -th time step of x .

► **Lemma 14** (Quantum circuits for computing the stopping times). *Let $\tilde{\tau}_{t'}$, $t' \in \{0, \dots, T\}$, be stopping times defined in Eq. (2). For all $t \in [T]$, given quantum query access to functions $\{z_{t'} : E \rightarrow \mathbb{R}\}_{t'=1}^T$ and $\{e_{t',k} : E \rightarrow \mathbb{R}\}_{t' \in [T-1], k \in [m]}$ in time \mathcal{T}_z and \mathcal{T}_e , respectively, and to the components of real vectors $\{\tilde{\alpha}_{t'} \in \mathbb{R}^m\}_{t'=t}^T$ in time $O(1)$, the following statements are true (let $e_{0,k}(x_0) := 1$):*

1. *There is a unitary W_t such that, in time $O(\mathcal{T}_z + m\mathcal{T}_e)$,*

$$\begin{cases} W_t |x\rangle |\tilde{\tau}_{t+1}(x)\rangle |\mathbf{0}\rangle^{\otimes 3} = |x\rangle |\tilde{\tau}_{t+1}(x)\rangle |z_t(x_t)\rangle |\tilde{\alpha}_t \cdot \vec{e}_t(x_t)\rangle |\tilde{\tau}_t(x)\rangle & \text{if } t \neq T, \\ W_t |x\rangle |\mathbf{0}\rangle = |x\rangle |T\rangle & \text{if } t = T. \end{cases}$$

2. *There is a unitary $V_t^{(k)}$ such that $V_t^{(k)} |x\rangle |\tilde{\tau}_t(x)\rangle |\mathbf{0}\rangle = |x\rangle |\tilde{\tau}_t(x)\rangle |z_{\tilde{\tau}_t(x)} e_{t-1,k}(x_{t-1})\rangle$, for $k \in [m]$, in time $O(T \log(T) \mathcal{T}_z + \mathcal{T}_e)$.*

3. *The unitary $C_t^{(k)} := W_T^\dagger \dots W_{t+1}^\dagger W_t^\dagger V_t^{(k)} W_t W_{t+1} \dots W_T$ is such that $C_t^{(k)} |x\rangle |\mathbf{0}\rangle^{\otimes (3(T-t)+2)} = |x\rangle |z_{\tilde{\tau}_t(x)} e_{t-1,k}(x_{t-1})\rangle |\mathbf{0}\rangle^{\otimes (3(T-t)+1)}$, for $k \in [m]$, in time $O(T(\log(T) \mathcal{T}_z + m\mathcal{T}_e))$.*

Proof. We start with the first statement. The existence of W_T is trivial. Assume $t \in [T-1]$, then, with one query access to function oracle z_t and m query accesses to function oracle $e_{t,k}$ for $k \in [m]$, we can perform

$$|x\rangle |\mathbf{0}\rangle |\mathbf{0}\rangle \mapsto |x\rangle |z_t(x_t)\rangle |\mathbf{0}\rangle \mapsto |x\rangle |z_t(x_t)\rangle |\vec{e}_t(x_t)\rangle.$$

By using access to the m elements of $\tilde{\alpha}_t$, and $O(m)$ multiplications and additions, we can compute the inner product of $\tilde{\alpha}_t \cdot \vec{e}_t(x_t)$ in superposition over x_t , as

$$|x\rangle |\vec{e}_t(x_t)\rangle |\mathbf{0}\rangle \mapsto |x\rangle |\vec{e}_t(x_t)\rangle |\tilde{\alpha}_t \cdot \vec{e}_t(x_t)\rangle.$$

Comparing between $z_t(x_t)$ and $\tilde{\alpha}_t \cdot \vec{e}_t(x_t)$ in constant time, we can compute $\tilde{\tau}_t(x)$ according to Eq. (2), and hence obtain

$$|x\rangle |\tilde{\tau}_{t+1}(x)\rangle |z_t(x_t)\rangle |\tilde{\alpha}_t \cdot \vec{e}_t(x_t)\rangle |\mathbf{0}\rangle \mapsto |x\rangle |\tilde{\tau}_{t+1}(x)\rangle |z_t(x_t)\rangle |\tilde{\alpha}_t \cdot \vec{e}_t(x_t)\rangle |\tilde{\tau}_t(x)\rangle.$$

Uncomputing the intermediate steps leads to the desired operation. The total runtime is $O(\mathcal{T}_z + m\mathcal{T}_e + m + m + 1) = O(\mathcal{T}_z + m\mathcal{T}_e)$.

Regarding the second statement, we require a few circuits which can be constructed once as a pre-processing step. First, we prepare the input for the payoff functions in an ancillary register, where the input depends on the content of the register $|\tilde{\tau}_t(x)\rangle$. For this step, we prepare a conditional copy quantum circuit V_{copy} which operates as $|x\rangle |\tilde{\tau}_t(x)\rangle |\mathbf{0}\rangle \rightarrow |x\rangle |\tilde{\tau}_t(x)\rangle |x_{\tilde{\tau}_t(x)}\rangle$, where the register $|x_{\tilde{\tau}_t(x)}\rangle$ stores the $\tilde{\tau}_t(x)$ -th step of the path x . This circuit operates in time given by the size of the registers of at most $O(T \log(T))$. Second, from the access to the different payoff functions we construct access to the functions in superposition of the time parameter. By assumption, we are given quantum circuits $V_{z_{t'}}$ for $t' \in [T]$. From these quantum circuits we construct the controlled circuit $V_{\text{select}} := \sum_{t'=1}^T |t'\rangle \langle t'| \otimes V_{z_{t'}}$, which consists of the controlled versions of the circuits $V_{z_{t'}}$ and has a runtime of $O(T \log(T) \mathcal{T}_z)$ [10]. Now, given $|x\rangle |\tilde{\tau}_t(x)\rangle$, with one application of V_{copy} and one application of V_{select} , we obtain $z_{\tilde{\tau}_t(x)}$, i.e, the payoff evaluated at $x_{\tilde{\tau}_t(x)}$, as

$$|x\rangle |\tilde{\tau}_t(x)\rangle |\mathbf{0}\rangle |\mathbf{0}\rangle \mapsto |x\rangle |\tilde{\tau}_t(x)\rangle |x_{\tilde{\tau}_t(x)}\rangle |\mathbf{0}\rangle \mapsto |x\rangle |\tilde{\tau}_t(x)\rangle |x_{\tilde{\tau}_t(x)}\rangle |z_{\tilde{\tau}_t(x)}\rangle.$$

Using V_{copy} again we uncompute the third register. One query to the function oracle $e_{t-1,k}$ obtains $e_{t-1,k}(x_{t-1})$ and by multiplication we obtain,

$$|x\rangle |\tilde{\tau}_t(x)\rangle |z_{\tilde{\tau}_t(x)}\rangle |e_{t-1,k}(x_{t-1})\rangle |\mathbf{0}\rangle \mapsto |x\rangle |\tilde{\tau}_t(x)\rangle |z_{\tilde{\tau}_t(x)}\rangle |e_{t-1,k}(x_{t-1})\rangle |z_{\tilde{\tau}_t(x)} e_{t-1,k}(x_{t-1})\rangle.$$

We uncompute the third and fourth registers using the given circuits to obtain the desired operation. The total runtime is $O(T \log(T) \mathcal{T}_z + \mathcal{T}_e)$.

Finally, for the third statement, it is not hard to see that (let $\tilde{\tau}_T := T$)

$$\begin{aligned} C_t^{(k)} |x\rangle |\mathbf{0}\rangle^{\otimes (3(T-t)+2)} &= W_T^\dagger \dots W_t^\dagger V_t^{(k)} |x\rangle |T\rangle |\mathbf{0}\rangle \bigotimes_{j=t}^{T-1} |z_j(x_j)\rangle |\tilde{\alpha}_j \cdot \tilde{e}_j(x_j)\rangle |\tilde{\tau}_j(x)\rangle \\ &= W_T^\dagger \dots W_t^\dagger |x\rangle |T\rangle |z_{\tilde{\tau}_t(x)} e_{t-1,k}(x_{t-1})\rangle \bigotimes_{j=t}^{T-1} |z_j(x_j)\rangle |\tilde{\alpha}_j \cdot \tilde{e}_j(x_j)\rangle |\tilde{\tau}_j(x)\rangle \\ &= |x\rangle |z_{\tilde{\tau}_t(x)} e_{t-1,k}(x_{t-1})\rangle |\mathbf{0}\rangle^{\otimes (3(T-t)+1)}. \end{aligned}$$

From the two previous statements, the runtime of $C_t^{(k)}$ is $2(T-t+1)O(\mathcal{T}_z + m\mathcal{T}_e) + O(T \log(T) \mathcal{T}_z + \mathcal{T}_e) = O(T(\log(T) \mathcal{T}_z + m\mathcal{T}_e))$. \blacktriangleleft

3.2 The algorithm

We present our quantum LSM algorithm in Algorithm 2. It computes the expectations $\mathbb{E}[e_{t,k}(X_t) e_{t,l}(X_t)]$ (for the matrices $\{A_t\}_{t=1}^{T-1}$), $\mathbb{E}[Z_{\tilde{\tau}_{t+1}} \tilde{e}_t(X_t)]$ and $\mathbb{E}[Z_{\tilde{\tau}_1}]$ using Theorem 13 instead of drawing random samples. Recall that by definition $Z_{\tilde{\tau}_{t+1}} = z_{\tilde{\tau}_{t+1}(\mathbf{x}_{t+1})}(X_{\tilde{\tau}_{t+1}(\mathbf{x}_{t+1})})$, i.e., both the optimal stopping time and the payoff depend on the path of the Markov chain.

As previously mentioned, it follows the classical version in Algorithm 1. However, the dynamic programming is not solved separately along different sampled paths, but in superposition along all possible stochastic processes. More specifically, at any given time t , the dynamic programming is solved in a backward fashion from time T to $t+1$ by constructing a unitary that prepares the approximate stopping times $\tilde{\tau}_{t+1}$ in superposition via the mapping $|x\rangle |\mathbf{0}\rangle \mapsto |x\rangle |\tilde{\tau}_{t+1}(x)\rangle$ for all $x \in E^T$. Such unitary is constructed (Lemma 14) using the values of all stopping times $\tilde{\tau}_{t+1}$ calculated so far in the dynamic program and allows access to the quantity $Z_{\tilde{\tau}_{t+1}}$, which in turn is used in the quantum subroutines for Monte Carlo to extract expectation values $\mathbb{E}[Z_{\tilde{\tau}_{t+1}} | X_t]$ that make up the vector b_t . The matrices $\{A_t\}_{t=1}^{T-1}$, in turn, are computed in an entrywise fashion at the start of the algorithm by using quantum access to the functions $e_{t,k}(x_t) e_{t,l}(x_t)$. In hold of the approximations \tilde{A}_t and \tilde{b}_t to A_t and b_t , respectively, the vector $\tilde{\alpha}_t = \tilde{A}_t^{-1} \tilde{b}_t$ is then computed classically and used to continue the dynamic programming at the next time step t when solving $\tilde{\tau}_t(x) = t \mathbf{1}\{z_t(x_t) \geq \tilde{\alpha}_t \cdot \tilde{e}_t(x_t)\} + \tilde{\tau}_{t+1}(x) \mathbf{1}\{z_t(x_t) < \tilde{\alpha}_t \cdot \tilde{e}_t(x_t)\}$ from time T to time t in superposition. Such procedure is repeated until $t=1$, when the optimal stopping time $\tilde{\tau}_1$ can be computed in superposition and thus the quantity $\sup_\tau \mathbb{E}[Z_\tau]$ can finally be approximated by $\max\{Z_0, \tilde{Z}_{\tilde{\tau}_1}\}$. We note that the procedure of approximating a matrix A_t and a vector b_t entrywise via quantum algorithms for Monte Carlo followed by the classical computation of $\tilde{\alpha}_t = \tilde{A}_t^{-1} \tilde{b}_t$ was already used in [50]. We also note that, unlike the classical LSM, our quantum algorithm requires redoing all previous dynamic programming steps before a given time t in order to progress into the next time step $t-1$. The final procedure involves $O(T^2)$ time steps instead of $O(T)$.

■ **Algorithm 2** Quantum LSM algorithm for optimal stopping problem.

Input: Parameters $\delta \in (0, 1)$, $\epsilon > 0$. Quantum sampling access to Markov chain $(X_t)_{t=1}^T$ defined on a finite sample space Ω and with finite state space $E \subseteq \mathbb{R}^d$. Quantum query access to $\{z_t : E \rightarrow \mathbb{R}\}_{t=1}^T$ and linearly independent functions $\{e_{t,k} : E \rightarrow \mathbb{R}\}_{k=1}^m$ for $t \in [T-1]$. Let $L := \max_{t \in [T-1], k \in [m]} \|e_{t,k}\|_{L^2(\rho_t)}$ and $R := \max_{t \in [T]} \|z_t\|_u$.

- 1: $\delta_A := \delta/(4Tm^2)$, $\delta_b := \delta/(4Tm)$, $\epsilon_A := \epsilon/m$ and $\epsilon_b := \epsilon/\sqrt{m}$.
- 2: Construct quantum access and controlled rotation access to $e_{t,k}e_{t,l}$, $\forall k, l \in [m]$, $t \in [T-1]$, with quantum query access to $e_{t,j}$, quantum circuits for multiplication and Fact 12.
- 3: Compute $\{\tilde{A}_t\}_{t=1}^{T-1}$ by calling **QMonteCarlo** $(e_{t,k}(X_t)e_{t,l}(X_t), \epsilon_A, \delta_A, L^2)$ for $k, l \in [m]$.
- 4: Compute the inverses $\{\tilde{A}_t^{-1}\}_{t=1}^{T-1}$.
- 5: Prepare unitary W_T s.t. $W_T|x\rangle|\mathbf{0}\rangle = |x\rangle|\tilde{\tau}_T(x)\rangle$, where $\tilde{\tau}_T(x) = T$ for all $x \in E^T$.
- 6: **for** $t = T$ to 2 **do**
- 7: **if** $t \neq T$ **then**
- 8: Prepare unitary W_t s.t. $W_t|x\rangle|\tilde{\tau}_{t+1}(x)\rangle|\mathbf{0}\rangle^{\otimes 3} = |x\rangle|\tilde{\tau}_{t+1}(x)\rangle|z_t(x_t)\rangle|\tilde{\alpha}_t \cdot \vec{e}_t(x_t)\rangle|\tilde{\tau}_t(x)\rangle$ for any $\tilde{\tau}_{t+1}(x) \in [T]$ (Lemma 14).
- 9: **end if**
- 10: Prepare unitaries $\{V_t^{(k)}\}_{k=1}^m$ s.t. $V_t^{(k)}|x\rangle|\tilde{\tau}_t(x)\rangle|\mathbf{0}\rangle = |x\rangle|\tilde{\tau}_t(x)\rangle|z_{\tilde{\tau}_t(x)}e_{t-1,k}(x_{t-1})\rangle$ (Lemma 14).
- 11: Prepare unitary $W_T^\dagger \dots W_{t+1}^\dagger W_t^\dagger V_t^{(k)} W_t W_{t+1} \dots W_T$ for $k \in [m]$ (Lemma 14).
- 12: Construct quantum access to the controlled rotations of the functions $z_{\tilde{\tau}_t(x)}e_{t-1,k}(x_{t-1})$ (Fact 12).
- 13: Execute **QMonteCarlo** $(Z_{\tilde{\tau}_t} e_{t-1,k}(X_{t-1}), \epsilon_b, \delta_b, RL)$, for all $k \in [m]$, to compute \tilde{b}_{t-1} .
- 14: Compute the vector $\tilde{\alpha}_{t-1} = \tilde{A}_{t-1}^{-1} \tilde{b}_{t-1}$ classically.
- 15: **end for**
- 16: Prepare unitary W_1 s.t. $W_1|x\rangle|\tilde{\tau}_2(x)\rangle|\mathbf{0}\rangle^{\otimes 3} = |x\rangle|\tilde{\tau}_2(x)\rangle|z_1(x_1)\rangle|\tilde{\alpha}_1 \cdot \vec{e}_1(x_1)\rangle|\tilde{\tau}_1(x)\rangle$ for any $\tilde{\tau}_2(x) \in [T]$ (Lemma 14).
- 17: Prepare unitary V_1 s.t. $V_1|x\rangle|\tilde{\tau}_1(x)\rangle|\mathbf{0}\rangle = |x\rangle|\tilde{\tau}_1(x)\rangle|z_{\tilde{\tau}_1(x)}\rangle$ (Lemma 14).
- 18: Prepare unitary $W_T^\dagger \dots W_2^\dagger W_1^\dagger V_1 W_1 W_2 \dots W_T$ (Lemma 14).
- 19: Construct quantum access to the controlled rotations of the function $z_{\tilde{\tau}_1}$ (Fact 12).
- 20: Execute **QMonteCarlo** $(Z_{\tilde{\tau}_1}, \epsilon, \frac{\delta}{2}, R)$ to compute $\tilde{Z}_{\tilde{\tau}_1}$.
- 21: Output $\tilde{\mathcal{U}}_0 := \max\{Z_0, \tilde{Z}_{\tilde{\tau}_1}\}$.

3.3 Error analysis and complexity

In Appendix A we prove that the classical LSM algorithm and our proposed quantum LSM algorithm approximate the sought-after quantity \mathcal{U}_0 up to additive accuracy with high probability. Among several results, the following encapsulates the overall complexity of the classical and quantum LSM algorithms. For simplicity we assume that $\mathcal{T}_{\text{samp}}, \mathcal{T}_z, \mathcal{T}_e = O(1)$.

► **Theorem 15** (Informal version of Corollary 22). *Consider a set of linearly independent functions $\{e_{t,k} : E \rightarrow \mathbb{R}\}_{k=1}^m$ for each $t \in [T-1]$ and payoff functions $\{z_t : E \rightarrow \mathbb{R}\}_{t=0}^T$. Then, for $\delta \in (0, 1)$ and $\epsilon > 0$, the classical and quantum LSM algorithms output $\tilde{\mathcal{U}}_0$ such that*

$$\Pr \left[|\tilde{\mathcal{U}}_0 - \mathcal{U}_0| \geq 5^T \left(\epsilon + \max_{0 < t < T} \min_{a \in \mathbb{R}^m} \|a \cdot \vec{e}_t(X_t) - \mathbb{E}[Z_{\tau_{t+1}} | X_t]\|_{L^2(\rho_t)} \right) \right] \leq \delta$$

using time, respectively, $\tilde{O}\left(\frac{Tm^6}{\epsilon^2}\right)$ and $\tilde{O}\left(\frac{T^2m^4}{\epsilon}\right)$, up to polylog terms.

The error ϵ arises from the Monte Carlo subroutines and can be made smaller by increasing the calls to the quantum inputs (or to the number of sampled paths in the classical counterpart). Compared to the classical algorithm, the number of oracle calls is quadratically less in the quantum algorithm. The quantity $\min_{a \in \mathbb{R}^m} \|a \cdot \vec{e}_t(X_t) - \mathbb{E}[Z_{\tau_{t+1}} | X_t]\|_{L^2(\rho_t)}$ appearing in the theorem above is known as *approximation error*. This term arises from approximating the continuation values by the m expansion functions and is a deterministic quantity implicitly dependent on m and on smoothness properties of the continuation values.

In order to obtain a final additive accuracy ϵ_{final} for $\tilde{\mathcal{U}}_0$, we must resolve the implicit dependence of the approximation error on m . This is done by considering specific sets of expansion functions and assuming sufficiently good smoothness properties for the continuation values. More specifically, for each $t \in [T - 1]$ we consider functions $\{e_{t,k} : E \rightarrow \mathbb{R}\}_{k=1}^m$ that generate the space \mathcal{R}_q of all polynomials of degree at most q , so that $m = \binom{q+d}{d}$. We also assume that $\mathbb{E}[Z_{\tau_{t+1}} | X_t] \in C^n$, i.e., the continuation values are n -differentiable functions. Then it is possible to bound the approximation error $\min_{a \in \mathbb{R}^m} \|a \cdot \vec{e}_t(X_t) - \mathbb{E}[Z_{\tau_{t+1}} | X_t]\|_{L^2(\rho_t)}$ by using a Jackson-like inequality [47] and obtain the following result (see the arXiv version [28] for the full statement and proof).

► **Theorem 16.** *For each $t \in [T - 1]$ consider a set of linearly independent functions $\{e_{t,k} : E \rightarrow \mathbb{R}\}_{k=1}^m$ that spans the space \mathcal{R}_q with $m = \binom{q+d}{d}$ and consider payoff functions $\{z_t : E \rightarrow \mathbb{R}\}_{t=0}^T$. Assume that $\mathbb{E}[Z_{\tau_{t+1}} | X_t] \in C^n$ for all $t \in \{0, \dots, T - 1\}$, where $n \leq q$. Then, for $\delta \in (0, 1)$ and $\epsilon \geq 0$, if $q = \lceil (5^T/\epsilon)^{1/n} \rceil$, the classical and quantum LSM algorithms output $\tilde{\mathcal{U}}_0$ such that $\Pr[|\tilde{\mathcal{U}}_0 - \mathcal{U}_0| \geq \epsilon] \leq \delta$ using time, respectively, $\tilde{O}((5^T/\epsilon)^{2+6d/n})$ and $\tilde{O}((5^T/\epsilon)^{1+4d/n})$, up to polylog terms.*

If the continuation values are n -times differentiable, for $n = \Theta(\log(5^T/\epsilon)/\log \log(5^T/\epsilon))$, then we get the sought-after quadratic improvement from $\tilde{O}((5^T/\epsilon)^2)$ classical runtime to $\tilde{O}(5^T/\epsilon)$ quantum runtime, up to polylog terms. We briefly note that such smoothness conditions on the continuation values are not unusual in areas like finance. Indeed, the continuation values can even be in C^∞ in some models, e.g. Black-Scholes [34, 85].

Very recently, Miyamoto [62] proposed a quantum LSM algorithm based on Chebyshev interpolation through Chebyshev nodes and obtained $O(\epsilon^{-1} \log^d(1/\epsilon) \text{poly log log}(1/\epsilon))$ as a final complexity. Our approach, in contrast, is to project $\mathbb{E}[Z_{\tau_{t+1}} | X_t]$ onto a set of polynomials and is, for this reason, much more general. Moreover, our final result is a *time* complexity, while the result from [62] is a *query* complexity on the number of unitaries called by all quantum routines for Monte Carlo. Finally, Miyamoto [62] assumes that the continuation values are analytical functions, i.e., are in C^∞ , while we only need to assume $\mathbb{E}[Z_{\tau_{t+1}} | X_t] \in C^n$ for $n = \Theta(\log(5^T/\epsilon)/\log \log(5^T/\epsilon))$ in order to recover $\tilde{O}(\epsilon^{-1})$ up to polylog factors. One downside of our approach, though, is the presence of quantities that implicitly depend on the underlying Markov chain.

As just mentioned, the full results behind the informal theorems above involve parameters that depend on the underlying Markov chain such as the minimum singular value of the matrices A_t . In order to explicitly work these parameters out, we also study the case when the underlying Markov process follows Brownian motion or geometric Brownian motion and obtain a simplified version of our algorithm (see arXiv version [28]). In the case of Brownian motion, we choose Hermite polynomials as the functions $\{e_{t,k} : E \rightarrow \mathbb{R}\}_{k=1}^m$ for each $t \in [T - 1]$, since they are orthogonal under the probability measure underlying a Brownian motion. This means that the matrices A_t are just the identity. The final result is a mild reduction on the classical and quantum time complexities to $\tilde{O}((5^T/\epsilon)^{2+4d/n})$ and $\tilde{O}((5^T/\epsilon)^{1+7d/2n})$, respectively. For the geometric Brownian motion,

we pick suitable monomials that reduce the matrices A_t to Vandermonde matrices, whose minimum singular value can be bounded. We obtain the final classical and quantum complexities $e^{O((5^T/\epsilon)^{2/n})}(5^T/\epsilon)^{2+12d/n}$ and $e^{O((5^T/\epsilon)^{2/n})}(5^T/\epsilon)^{1+15d/2n}$, respectively. If the continuation values are again n -times differentiable for $n = \Theta(\log(5^T/\epsilon)/\log\log(5^T/\epsilon))$, then the classical and quantum complexities for the Brownian motion setting reduce to the usual $\tilde{O}(\epsilon^{-2})$ and $\tilde{O}(\epsilon^{-1})$, respectively, while, for the geometric Brownian motion, they reduce to $e^{O(\log^c(5^T/\epsilon))}(5^T/\epsilon)^2$ and $e^{O(\log^c(5^T/\epsilon))}(5^T/\epsilon)$ for any constant $0 < c < 1$. These results for the geometric Brownian motion are slightly weaker than the usual $\tilde{O}((5^T/\epsilon)^2)$ and $\tilde{O}(5^T/\epsilon)$, since the bound on the minimum singular value of the matrix A_t is very sensitive to the degree q of the chosen monomials.

4 Conclusions

In this work, we developed a new quantum algorithm for a stochastic optimal stopping problem (as in Theorem 4) with a quantum advantage in the runtime. This problem cannot be solved accurately by a single application of quantum algorithms for Monte Carlo [63, 41, 22, 21, 3]. Instead one must compute in superposition (and recursively) the stopping times as in Lemma 14, which is key to obtaining a quantum speedup. As the classical LSM algorithm can be used to solve a large variety of problems, our quantum LSM can also be used for problems in finance including insurance [53] and risk management [38], and for many optimization problems outside finance, such as quickest detection [82] and sequential Bayesian hypothesis testing [26]. Additionally, we believe that there are many other problems in, for example, dynamic programming, stochastic optimal stopping and optimal control where the interplay of function approximation and quantum subroutines for Monte Carlo could be used to design new quantum algorithms.

A few design choices of the quantum algorithm were guided by real problems where the classical algorithm is already used. Even though we took number of expansion functions $m = \text{poly}(5^T/\epsilon)$ in order to bound the approximation error, in practice one typically assumes m to be constant [57]. For big values of m , our algorithm could be modified in order to use quantum subroutines for inner product estimation, and reduce the complexity polynomially in m , but introducing a further ϵ dependence. Thus, further analysis is needed to understand the impact of the precision parameters on the runtime of these subroutines. Along these lines, we have chosen to invert the linear systems for finding α_t on a classical computer. A possible modification of our algorithm could output quantum states $|\alpha_t\rangle$ via HHL-like algorithms [43]. We also discussed how, under the hypothesis that the Markov chain is a Brownian motion or a Geometric Brownian motion, the matrices A_t can be expressed with a closed formula and their minimum singular value be bounded. This idea exhibits some similarity with the idea proposed in [56]. There, they leveraged quasi-regression algorithms and a particular choice of expansion functions [68], so to pre-compute the matrices A_t , and thus skip costly Monte Carlo estimators. Moreover, when considering a Brownian or a Geometric Brownian motion, the chosen functions $\{e_{t,k}\}_{k=1}^m$ had an explicit time dependence on t , but it is possible to transform the optimal stopping time problem behind American option pricing in a way that such dependence is suppressed and a single set of approximating functions is employed [15, 16]. We believe that such approach could improve our complexities. Finally, in our algorithm, we use quantum subroutines from [63], but could equivalently use the subroutines from [41, 22, 21]. Our template could be extended to quantum algorithms that are similar in spirit but are solving different problems [87].

Our final complexities have an exponential dependence on T , the number of time steps. We believe that such dependence, present in several past works [30, 93, 94, 95, 96, 62], is only a consequence of a loose error bound and could possibly be improved. Such hope is backed up by the ubiquitous employment of LSM algorithms for pricing American options in every day financial markets. We also note that a more careful error analysis would improve classical results as well, but, regardless, the quantum quadratic improvement would still be present. Finally, notice that it is always possible to compensate a reduction on the number of time steps with more accurate approximations for continuation values and similar quantities.

We stress the importance of fast quantum algorithms for optimal stopping problems. For American option pricing, the value of the payoff function could easily reach a few million dollars, and the additive precision ϵ could be of the order of 10^{-11} [4]. Given the level of specialization in classical algorithms and architectures for this specific problem, how and when our algorithm can find application in practice is left for future work.

References

- 1 Directive 2009/138/EC of the European Parliament and of the Council of 25 November 2009 on the taking-up and pursuit of the business of Insurance and Reinsurance (Solvency II), 2009.
- 2 Javier Alcazar, Vicente Leyton-Ortega, and Alejandro Perdomo-Ortiz. Classical versus quantum models in machine learning: insights from a finance application. *Machine Learning: Science and Technology*, 1(3):035003, 2020.
- 3 Dong An, Noah Linden, Jin-Peng Liu, Ashley Montanaro, Changpeng Shao, and Jiasu Wang. Quantum-accelerated multilevel Monte Carlo methods for stochastic differential equations in mathematical finance. *Quantum*, 5:481, 2021.
- 4 Leif B. G. Andersen, Mark Lake, and Dimitri Offengenden. High-performance American option pricing. *Journal of Computational Finance*, 20(1):39–87, 2016.
- 5 Anna Rita Bacinello, Enrico Biffis, and Pietro Millossovich. Pricing life insurance contracts with early exercise features. *Journal of computational and applied mathematics*, 233(1):27–35, 2009.
- 6 Panagiotis Kl. Barkoutsos, Giacomo Nannicini, Anton Robert, Ivano Tavernelli, and Stefan Woerner. Improving variational quantum optimization using CVaR. *Quantum*, 4:256, 2020.
- 7 Giovanni Barone-Adesi and Robert E. Whaley. Efficient analytic approximation of American option values. *the Journal of Finance*, 42(2):301–320, 1987.
- 8 Christian Bender and Jessica Steiner. Least-squares Monte Carlo for backward SDEs. In *Numerical methods in finance*, pages 257–289. Springer, 2012.
- 9 Alain Bensoussan and J.-L. Lions. *Applications of variational inequalities in stochastic control*. Elsevier, 2011.
- 10 Dominic W. Berry, Andrew M. Childs, Richard Cleve, Robin Kothari, and Rolando D. Somma. Simulating Hamiltonian dynamics with a truncated Taylor series. *Phys. Rev. Lett.*, 114:090502, 2015. doi:10.1103/PhysRevLett.114.090502.
- 11 Fischer Black and Myron Scholes. The pricing of options and corporate liabilities. *Journal of Political Economy*, 81:637–654, 1973.
- 12 Adam Bouland, Wim van Dam, Hamed Joorati, Iordanis Kerenidis, and Anupam Prakash. Prospects and challenges of quantum finance. *arXiv preprint*, 2020. arXiv:2011.06492.
- 13 Phelim P. Boyle. Options: a Monte Carlo approach. *Journal of Financial Economics*, 4(3):323–338, 1977.
- 14 Gilles Brassard, Peter Hoyer, Michele Mosca, and Alain Tapp. Quantum amplitude amplification and estimation. *Contemporary Mathematics*, 305:53–74, 2002.
- 15 Mark Broadie, Jérôme Detemple, Eric Ghysels, and Olivier Torrès. American options with stochastic dividends and volatility: A nonparametric investigation. *Journal of Econometrics*, 94(1-2):53–92, 2000.

- 16 Mark Broadie, Jérôme Detemple, Eric Ghysels, and Olivier Torrès. Nonparametric estimation of American options' exercise boundaries and call prices. *Journal of Economic Dynamics and Control*, 24(11-12):1829–1857, 2000.
- 17 Mark Broadie and Paul Glasserman. Monte Carlo methods for pricing high-dimensional American options: an overview. *Net Exposure*, 3:15–37, 1997.
- 18 Shouvanik Chakrabarti, Rajiv Krishnakumar, Guglielmo Mazzola, Nikitas Stamatopoulos, Stefan Woerner, and William J. Zeng. A threshold for quantum advantage in derivative pricing. *Quantum*, 5:463, 2021.
- 19 Don M. Chance. A synthesis of binomial option pricing models for lognormally distributed assets. *Journal of Applied Finance (Formerly Financial Practice and Education)*, 18(1), 2008.
- 20 Emmanuelle Clément, Damien Lamberton, and Philip Protter. An analysis of a least squares regression method for American option pricing. *Finance and Stochastics*, 6(4):449–471, 2002.
- 21 Arjan Cornelissen, Yassine Hamoudi, and Sofiene Jerbi. Near-optimal quantum algorithms for multivariate mean estimation. *arXiv preprint*, 2021. [arXiv:2111.09787](https://arxiv.org/abs/2111.09787).
- 22 Arjan Cornelissen and Sofiene Jerbi. Quantum algorithms for multivariate Monte Carlo estimation. *arXiv preprint*, 2021. [arXiv:2107.03410](https://arxiv.org/abs/2107.03410).
- 23 John C. Cox, Stephen A. Ross, and Mark Rubinstein. Option pricing: a simplified approach. *Journal of Financial Economics*, 7(3):229–263, 1979.
- 24 Gary Wayne Crosby. *Optimal multiple stopping: theory and applications*. PhD thesis, The University of North Carolina at Charlotte, 2017.
- 25 Samudra Dasgupta and Arnab Banerjee. Quantum annealing algorithm for expected shortfall based dynamic asset allocation. *arXiv preprint*, 2019. [arXiv:1909.12904](https://arxiv.org/abs/1909.12904).
- 26 Constantinos Daskalakis and Yasushi Kawase. Optimal stopping rules for sequential hypothesis testing. In *25th Annual European Symposium on Algorithms (ESA)*, 2017.
- 27 Georgios Dimitrakopoulos. Least-squares Monte Carlo simulation and high performance computing for Solvency II regulatory capital estimation. Master's thesis, The University of Manchester (United Kingdom), 2013.
- 28 João F. Doriguello, Alessandro Luongo, Jinge Bao, Patrick Rebentrost, and Miklos Santha. Quantum algorithm for stochastic optimal stopping problems with applications in finance. *arXiv preprint*, 2021. [arXiv:2111.15332](https://arxiv.org/abs/2111.15332).
- 29 Daniel J. Egger, Claudio Gambella, Jakub Marecek, Scott McFaddin, Martin Mevissen, Rudy Raymond, Andrea Simonetto, Stefan Woerner, and Elena Yndurain. Quantum computing for finance: state of the art and future prospects. *IEEE Transactions on Quantum Engineering*, 2020.
- 30 Daniel Egloff. Monte Carlo algorithms for optimal stopping and statistical learning. *The Annals of Applied Probability*, 15(2):1396–1432, 2005.
- 31 Hans Föllmer and Alexander Schied. *Stochastic finance*. de Gruyter, 2016.
- 32 Filipe Fontanela, Antoine Jacquier, and Mugad Oumgari. A quantum algorithm for linear PDEs arising in finance. *SIAM Journal on Financial Mathematics*, 12(4):SC98–SC114, 2021.
- 33 Michael C. Fu, Scott B. Laprise, Dilip B. Madan, Yi Su, and Rongwen Wu. Pricing American options: a comparison of Monte Carlo simulation approaches. *Journal of Computational Finance*, 4(3):39–88, 2001.
- 34 Stefan Gerhold. The Longstaff-Schwartz algorithm for Lévy models: results on fast and slow convergence. *The Annals of Applied Probability*, 21(2):589–608, 2011.
- 35 Emmanuel Gobet, Jean-Philippe Lemor, and Xavier Warin. A regression-based Monte Carlo method to solve backward stochastic differential equations. *The Annals of Applied Probability*, 15(3):2172–2202, 2005.
- 36 Emmanuel Gobet and Plamen Turkedjiev. Approximation of discrete BSDE using least-squares regression, November 2011. Technical report. URL: <https://hal.archives-ouvertes.fr/hal-00642685>.
- 37 Gene H. Golub and Charles F. Van Loan. *Matrix computations*, 2013.

- 38 Andrew Green. *XVA: Credit, Funding and Capital Valuation Adjustments*. John Wiley & Sons, 2015.
- 39 Lov Grover and Terry Rudolph. Creating superpositions that correspond to efficiently integrable probability distributions. *arXiv preprint*, 2002. [arXiv:quant-ph/0208112](https://arxiv.org/abs/quant-ph/0208112).
- 40 Eli Gutin. *Practical applications of large-scale stochastic control for learning and optimization*. PhD thesis, Massachusetts Institute of Technology, 2018.
- 41 Yassine Hamoudi. Quantum sub-Gaussian mean estimator. In *29th Annual European Symposium on Algorithms (ESA 2021)*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2021.
- 42 Jeong Yu Han and Patrick Rebentrost. Quantum advantage for multi-option portfolio pricing and valuation adjustments. *arXiv preprint*, 2022. [arXiv:2203.04924](https://arxiv.org/abs/2203.04924).
- 43 Aram W. Harrow, Avinatan Hassidim, and Seth Lloyd. Quantum algorithm for linear systems of equations. *Physical review letters*, 103(15):150502, 2009.
- 44 Steven Herbert. No quantum speedup with Grover-Rudolph state preparation for quantum Monte Carlo integration. *Physical Review E*, 103(6):063302, 2021.
- 45 Mark Hodson, Brendan Ruck, Hugh Ong, David Garvin, and Stefan Dulman. Portfolio rebalancing experiments using the quantum alternating operator ansatz. *arXiv preprint*, 2019. [arXiv:1911.05296](https://arxiv.org/abs/1911.05296).
- 46 Jacqueline Huang and Jong-Shi Pang. Option pricing and linear complementarity. Technical report, Cornell University, 2003.
- 47 Dunham Jackson. A general class of problems in approximation. *American Journal of Mathematics*, 46(4):215–234, 1924.
- 48 Patrick Jaillet, Damien Lambertson, and Bernard Lapeyre. Variational inequalities and the pricing of American options. *Acta Applicandae Mathematica*, 21(3):263–289, 1990.
- 49 Kazuya Kaneko, Koichi Miyamoto, Naoyuki Takeda, and Kazuyoshi Yoshino. Quantum pricing with a smile: implementation of local volatility model on quantum computer. *arXiv preprint*, 2020. [arXiv:2007.01467](https://arxiv.org/abs/2007.01467).
- 50 Kazuya Kaneko, Koichi Miyamoto, Naoyuki Takeda, and Kazuyoshi Yoshino. Linear regression by quantum amplitude estimation and its extension to convex optimization. *Phys. Rev. A*, 104:022430, 2021. [doi:10.1103/PhysRevA.104.022430](https://doi.org/10.1103/PhysRevA.104.022430).
- 51 In Joon Kim. The analytic valuation of American options. *The Review of Financial Studies*, 3(4):547–572, 1990.
- 52 Michael Kohler. A review on regression-based Monte Carlo methods for pricing American options. In *Recent developments in applied probability and statistics*, pages 37–58. Springer, 2010.
- 53 Anne-Sophie Krah, Zoran Nikolić, and Ralf Korn. A least-squares Monte Carlo framework in proxy modeling of life insurance companies. *Risks*, 6(2):62, 2018.
- 54 Jean-Philippe Lemor, Emmanuel Gobet, and Xavier Warin. Rate of convergence of an empirical regression method for solving generalized backward stochastic differential equations. *Bernoulli*, 12(5):889–916, 2006.
- 55 Xiaofei Li, Yi Wu, Quanxin Zhu, Songbo Hu, and Chuan Qin. A regression-based Monte Carlo method to solve two-dimensional forward backward stochastic differential equations. *Advances in Difference Equations*, 2021(1):1–13, 2021.
- 56 Chen Liu, Henry Schellhorn, and Qidi Peng. American option pricing with regression: convergence analysis. *International Journal of Theoretical and Applied Finance*, 22(08):1950044, 2019.
- 57 Francis A. Longstaff and Eduardo S. Schwartz. Valuing American options by simulation: a simple least-squares approach. *The review of financial studies*, 14(1):113–147, 2001.
- 58 Gabriel Marin-Sancheza, Javier Gonzalez-Conde, and Mikel Sanz. Quantum algorithms for approximate function loading. *arXiv preprint*, 2021. [arXiv:2111.07933](https://arxiv.org/abs/2111.07933).
- 59 Ana Martin, Bruno Candelas, Ángel Rodríguez-Rozas, José D. Martín-Guerrero, Xi Chen, Lucas Lamata, Román Orús, Enrique Solano, and Mikel Sanz. Toward pricing financial derivatives with an IBM quantum computer. *Physical Review Research*, 3(1):013167, 2021.

- 60 Henry P. McKean Jr. A free boundary problem for the heat equation arising from a problem of mathematical economics. *Industrial Management Review*, 6:32–39, 1965.
- 61 Robert C. Merton. Theory of rational option pricing. *The Bell Journal of Economics and Management Science*, 4(1):141–183, 1973.
- 62 Koichi Miyamoto. Bermudan option pricing by quantum amplitude estimation and Chebyshev interpolation. *arXiv preprint*, 2021. [arXiv:2108.09014](https://arxiv.org/abs/2108.09014).
- 63 Ashley Montanaro. Quantum speedup of Monte Carlo methods. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 471(2181):20150301, 2015.
- 64 Alexey Muravlev and Mikhail Zhitlukhin. A Bayesian sequential test for the drift of a fractional Brownian motion. *Advances in Applied Probability*, 52(4):1308–1324, 2020.
- 65 Jacques Neveu. *Discrete-parameter martingales*. Elsevier, 1975.
- 66 Michael A. Nielsen and Isaac L. Chuang. *Quantum Computation and Quantum Information*. Cambridge University Press, 2010. doi:10.1017/CB09780511976667.
- 67 Roman Orus, Samuel Mugel, and Enrique Lizaso. Quantum computing for finance: overview and prospects. *Reviews in Physics*, 4:100028, 2019.
- 68 Art B. Owen. Assessing linearity in high dimensions. *The Annals of Statistics*, 28(1):1–19, 2000.
- 69 Antoon Pelsser and Janina Schweizer. The difference between lsmc and replicating portfolio in insurance liability modeling. *European actuarial journal*, 6(2):441–494, 2016.
- 70 Goran Peskir and Albert Shiryaev. *Optimal stopping and free-boundary problems*. Springer, 2006.
- 71 Warren B. Powell. A unified framework for stochastic optimization. *European Journal of Operational Research*, 275(3):795–821, 2019.
- 72 Santosh Kumar Radha. Quantum option pricing using wick rotated imaginary time evolution. *arXiv preprint*, 2021. [arXiv:2101.04280](https://arxiv.org/abs/2101.04280).
- 73 Sergi Ramos-Calderer, Adrián Pérez-Salinas, Diego García-Martín, Carlos Bravo-Prieto, Jorge Cortada, Jordi Planaguma, and José I. Latorre. Quantum unary approach to option pricing. *Physical Review A*, 103(3):032414, 2021.
- 74 Patrick Rebentrost, Brajesh Gupta, and Thomas R. Bromley. Quantum computational finance: Monte Carlo pricing of financial derivatives. *Phys. Rev. A*, 98:022321, 2018.
- 75 Patrick Rebentrost and Seth Lloyd. Quantum computational finance: quantum algorithm for portfolio optimization. *arXiv preprint*, 2018. [arXiv:1811.03975](https://arxiv.org/abs/1811.03975).
- 76 Patrick Rebentrost, Miklos Santha, and Siyi Yang. Quantum alphasatron. *arXiv preprint*, 2021. [arXiv:2108.11670](https://arxiv.org/abs/2108.11670).
- 77 Richard J. Rendleman. Two-state option pricing. *The Journal of Finance*, 34(5):1093–1110, 1979.
- 78 Leonard C. G. Rogers. Monte Carlo valuation of American options. *Mathematical Finance*, 12(3):271–286, 2002.
- 79 Joose Mikko Juhani Sauli. On the suitability of the Longstaff-Schwartz term structure model for modelling the cost of government debt. Master’s thesis, Helsingfors Universitet, 2013.
- 80 William F. Sharpe, Gordon J. Alexander, and Jeffrey W. Bailey. *Investments*. Prentice-Hall, 1999.
- 81 Albert N. Shiryaev. *Optimal stopping rules*, volume 8. Springer Science & Business Media, 2007.
- 82 Albert N. Shiryaev. Quickest detection problems: fifty years later. *Sequential Analysis*, 29(4):345–385, 2010.
- 83 Nikitas Stamatopoulos, Daniel J. Egger, Yue Sun, Christa Zoufal, Raban Iten, Ning Shen, and Stefan Woerner. Option pricing using quantum computers. *arXiv preprint*, 2019. [arXiv:1905.02666](https://arxiv.org/abs/1905.02666).
- 84 Nikitas Stamatopoulos, Guglielmo Mazzola, Stefan Woerner, and William J. Zeng. Towards quantum advantage in financial market risk using quantum gradient algorithms. *arXiv preprint*, 2021. [arXiv:2111.12509](https://arxiv.org/abs/2111.12509).

- 85 Peter Tankov, Ekaterina Voltchkova, and Rama Cont. Option pricing models with jumps: integro-differential equations and inverse problems. In *European Congress on Computational Methods in Applied Sciences and Engineering (ECCOMAS 2004)*, 2004.
- 86 Elliot Tonkes and Dharma Lesmono. A Longstaff and Schwartz approach to the early election problem. *Advances in Decision Sciences*, 2012, 2012.
- 87 John N. Tsitsiklis and Benjamin Van Roy. Regression methods for pricing complex American-style options. *IEEE Transactions on Neural Networks*, 12(4):694–703, 2001.
- 88 Pierre Van Moerbeke. On optimal stopping and free boundary problems. *Archive for Rational Mechanics and Analysis*, 60(2):101–148, 1976.
- 89 Almudena Carrera Vazquez and Stefan Woerner. Efficient state preparation for quantum amplitude estimation. *Physical Review Applied*, 15(3):034027, 2021.
- 90 Abraham Wald. *Sequential Analysis*. John Wiley and Sons, 1st edition edition, 1947.
- 91 Pawel Wocjan, Chen-Fu Chiang, Daniel Nagaj, and Anura Abeyesinghe. Quantum algorithm for approximating partition functions. *Phys. Rev. A*, 80:022340, 2009.
- 92 Stefan Woerner and Daniel J. Egger. Quantum risk analysis. *arXiv preprint*, 2018. arXiv:1806.06893.
- 93 Daniel Z. Zanger. Convergence of a least-squares Monte Carlo algorithm for bounded approximating sets. *Applied Mathematical Finance*, 16(2):123–150, 2009.
- 94 Daniel Z. Zanger. Quantitative error estimates for a least-squares Monte Carlo algorithm for American option pricing. *Finance and Stochastics*, 17(3):503–534, 2013.
- 95 Daniel Z. Zanger. Convergence of a least-squares Monte Carlo algorithm for American option pricing with dependent sample data. *Mathematical Finance*, 28(1):447–479, 2018.
- 96 Daniel Z. Zanger. General error estimates for the Longstaff-Schwartz least-squares Monte Carlo algorithm. *Mathematics of Operations Research*, 45(3):923–946, 2020.

A Error Analysis and Complexity

In what follows, given $b \in \mathbb{R}^m$ and $A \in \mathbb{R}^{m \times m}$ for some $m \in \mathbb{N}$, let $\|b\|_2 := \sqrt{\sum_{i=1}^m b_i^2}$ and $\|A\|_2 := \sigma_{\max}(A)$ be the vector and matrix norms, respectively, where $\sigma_{\max}(A)$ is the maximum singular value of A . We shall denote by $\sigma_{\min}(A)$ the minimum singular value of A . Let ω_* denote the matrix multiplication exponent. Moreover, recall the uniform norm $\|f\|_u = \sup\{|f(s)| : s \in E\}$ for $f : E \rightarrow \mathbb{R}$.

We shall analyze the approximation error and complexity from Algorithm 2. In order to do so, we will need the following result from [93, 96] (already modified to our notation) that bounds the error between the exact continuation values $\mathbb{E}[Z_{\tau_{t+1}} | X_t]$ and their approximation $\tilde{\alpha}_t \cdot \tilde{e}_t(X_t)$ in terms of the error between the continuation values $\mathbb{E}[Z_{\tilde{\tau}_{k+1}} | X_k]$ evaluated on the approximated stopping times $\tilde{\tau}_{k+1}$ and $\tilde{\alpha}_k \cdot \tilde{e}_k(X_k)$ for $k \in \{t, \dots, T-1\}$. Recall the image probability measures ρ_t in $E \subseteq \mathbb{R}^d$ induced by each element X_t , $t \in \{0, \dots, T\}$.

► **Lemma 17** ([93, Lemma 2.2]). *For each $t \in \{0, \dots, T-1\}$, we have*

$$\begin{aligned} \|\tilde{\alpha}_t \cdot \tilde{e}_t(X_t) - \mathbb{E}[Z_{\tau_{t+1}} | X_t]\|_{L^2(\rho_t)} &\leq 2 \sum_{k=t}^{T-1} \|\tilde{\alpha}_k \cdot \tilde{e}_k(X_k) - \mathbb{E}[Z_{\tilde{\tau}_{k+1}} | X_k]\|_{L^2(\rho_k)}, \\ \|\mathbb{E}[Z_{\tilde{\tau}_{t+1}} | X_t] - \mathbb{E}[Z_{\tau_{t+1}} | X_t]\|_{L^2(\rho_t)} &\leq 2 \sum_{k=t+1}^{T-1} \|\tilde{\alpha}_k \cdot \tilde{e}_k(X_k) - \mathbb{E}[Z_{\tilde{\tau}_{k+1}} | X_k]\|_{L^2(\rho_k)}, \end{aligned}$$

where $\tilde{\alpha}_0 \cdot \tilde{e}_0(X_0) := \tilde{Z}_{\tilde{\tau}_1}$ approximates $\mathbb{E}[Z_{\tilde{\tau}_1}]$.

We will also need the following technical result on the sensitivity of square systems.

► **Theorem 18** ([37, Theorem 2.6.2]). *Let $Ax = b$ and $\tilde{A}\tilde{x} = \tilde{b}$, where $A, \tilde{A} \in \mathbb{R}^{d \times d}$ and $b, \tilde{b} \in \mathbb{R}^d$, with $b \neq 0$. Suppose that $\|A - \tilde{A}\|_2 \leq \epsilon_A$ and $\|b - \tilde{b}\|_2 \leq \epsilon_b$. If $\epsilon_A \leq \sigma_{\min}(A)/2$, where $\sigma_{\min}(A)$ is the minimum singular value of A , then*

$$\|x - \tilde{x}\|_2 \leq \frac{2}{\sigma_{\min}(A)} \left(\frac{\epsilon_A \|b\|_2}{\sigma_{\min}(A)} + \epsilon_b \right).$$

We are now able to state a central theorem for our quantum LSM algorithm.

► **Theorem 19.** *Within the setting of Algorithm 2 with input parameters δ and ϵ , let $T\mathcal{T}_{\text{samp}}$ be the sampling cost of the Markov chain and consider a set of linearly independent functions $\{e_{t,k} : E \rightarrow \mathbb{R}\}_{k=1}^m$ for each $t \in [T-1]$ with $L := \max_{t \in [T-1], k \in [m]} \|e_{t,k}\|_{L^2(\rho_t)}$ and query cost \mathcal{T}_e . Also consider $\{z_t : E \rightarrow \mathbb{R}\}_{t=0}^T$ with $R := \max_{t \in [T]} \|z_t\|_u < \infty$ and query cost \mathcal{T}_z . Moreover, let $\sigma_{\min} := \min_{t \in [T-1]} \sigma_{\min}(A_t) > 0$. Assume that $\sqrt{m}RL/\sigma_{\min} \geq 1$ and define $\mathcal{T}_{\text{total}} := \mathcal{T}_{\text{samp}} + \mathcal{T}_z + \mathcal{T}_e$. Then, for any $\delta \in (0, 1)$ and $\epsilon \in (0, \sigma_{\min}/2]$, Algorithm 2 outputs $\tilde{\mathcal{U}}_0$ such that*

$$\Pr \left[\|\tilde{\mathcal{U}}_0 - \mathbb{E}[Z_{\tau_0}]\| \geq \frac{8T\epsilon mRL^2}{\sigma_{\min}^2} + 2 \sum_{t=1}^{T-1} \min_{a \in \mathbb{R}^m} \|a \cdot \vec{e}_t(X_t) - \mathbb{E}[Z_{\tilde{\tau}_{t+1}} | X_t]\|_{L^2(\rho_t)} \right] \leq \delta \quad (11)$$

in time

$$O \left(\frac{T^2 m^3}{\epsilon} \mathcal{T}_{\text{total}} L(L+R) \log(T) \log(Tm^2/\delta) \log^{3/2}(mL(L+R)/\epsilon) \log \log(mL(L+R)/\epsilon) \right).$$

Proof. Set $b_t := \mathbb{E}[Z_{\tilde{\tau}_{t+1}} \vec{e}_t(X_t)]$. Recall that $\epsilon_A := \epsilon/m$, $\epsilon_b := \epsilon/\sqrt{m}$, $\delta_A := \delta/(4Tm^2)$ and $\delta_b := \delta/(4Tm)$. We start by computing the complexity of the algorithm. We first note the bounds $\|b_t\|_2 \leq R \|\mathbb{E}[\vec{e}_t(X_t)]\|_2 \leq \sqrt{m}RL$ and $\|A_t\|_2 \leq m \max_{k,l} \|\mathbb{E}[e_{t,k}(X_t)e_{t,l}(X_t)]\| \leq mL^2$ for all $t \in [T-1]$. The computation of all the entries of the matrices $\{A_t\}_{t=1}^{T-1}$ requires time

$$O \left(\frac{Tm^2}{\epsilon_A} L^2 (T\mathcal{T}_{\text{samp}} + \mathcal{T}_e) \log(1/\delta_A) \log^{3/2} \left(\frac{L^2}{\epsilon_A} \right) \log \log \left(\frac{L^2}{\epsilon_A} \right) \right),$$

by calling $\text{QMonteCarlo}(e_{t,k}(X_t)e_{t,l}(X_t), \epsilon_A, \delta_A, L^2)$ from Theorem 13 for $t \in [T-1]$ and $k, l \in [m]$. The computation of all b_t uses time

$$O \left(\frac{Tm}{\epsilon_b} RL (T\mathcal{T}_{\text{samp}} + T(\log(T)\mathcal{T}_z + m\mathcal{T}_e)) \log(1/\delta_b) \log^{3/2} \left(\frac{RL}{\epsilon_b} \right) \log \log \left(\frac{RL}{\epsilon_b} \right) \right),$$

by calling $\text{QMonteCarlo}(Z_{\tilde{\tau}_t} e_{t-1,k}(X_{t-1}), \epsilon_b, \delta_b, RL)$ from Theorem 13 for $t \in \{2, \dots, T\}$ and $k \in [m]$. Note that the term $T(\log(T)\mathcal{T}_z + m\mathcal{T}_e)$ comes from using the unitaries $C_t^{(k)}$ in QMonteCarlo , each with cost $O(T(\log(T)\mathcal{T}_z + m\mathcal{T}_e))$ according to Lemma 14. Computing $\mathbb{E}[Z_{\tilde{\tau}_1}]$ requires time

$$O \left(\frac{R}{\epsilon} (T\mathcal{T}_{\text{samp}} + T(\log(T)\mathcal{T}_z + m\mathcal{T}_e)) \log(1/\delta) \log^{3/2} \left(\frac{R}{\epsilon} \right) \log \log \left(\frac{R}{\epsilon} \right) \right),$$

by calling $\text{QMonteCarlo}(Z_{\tilde{\tau}_1}, \epsilon, \frac{\delta}{2}, R)$ from Theorem 13 and where the term $T(\log(T)\mathcal{T}_z + m\mathcal{T}_e)$ again comes from the unitaries $C_t^{(k)}$ in QMonteCarlo . The classical computation of $\{\tilde{A}_t^{-1}\}_{t=1}^{T-1}$ and $\{\tilde{\alpha}_t = \tilde{A}_t^{-1}\tilde{b}_t\}_{t=1}^{T-1}$ requires time $O(Tm^{\omega_*})$, where $2 \leq \omega_* < 3$. Hence, by keeping the largest terms of each complexity, the final complexity is upper-bounded by

$$O \left(\frac{T^2 m^3}{\epsilon} \mathcal{T}_{\text{total}} L(L+R) \log(T) \log(Tm^2/\delta) \log^{3/2}(mL(L+R)/\epsilon) \log \log(mL(L+R)/\epsilon) \right).$$

We now move to the error analysis. Fix $t \in [T - 1]$. We start by bounding the error $\|\tilde{\alpha}_t - \alpha_t\|_2$ between $\alpha_t = A_t^{-1}b_t$ and $\tilde{\alpha}_t = \tilde{A}_t^{-1}\tilde{b}_t$. By using **QMonteCarlo** from Theorem 13 we approximate each entry of A_t and b_t as $|(A_t)_{jl} - (\tilde{A}_t)_{jl}| \leq \epsilon/m$ and $|(b_t)_j - (\tilde{b}_t)_j| \leq \epsilon/\sqrt{m}$ for all $j, l \in [m]$. All approximations hold with probability at least $1 - \delta/2T$ by the union bound. This means that, with probability at least $1 - \delta/2T$,

$$\|A_t - \tilde{A}_t\|_2 \leq \sqrt{\sum_{j,l=1}^m |(A_t)_{jl} - (\tilde{A}_t)_{jl}|^2} \leq \epsilon, \quad \|b_t - \tilde{b}_t\|_2 = \sqrt{\sum_{j=1}^m |(b_t)_j - (\tilde{b}_t)_j|^2} \leq \epsilon.$$

According to Theorem 18, we obtain

$$\|\tilde{\alpha}_t - \alpha_t\|_2 \leq \frac{2\epsilon}{\sigma_{\min}(A_t)} \left(1 + \frac{\|b_t\|_2}{\sigma_{\min}(A_t)}\right) \leq \frac{4\epsilon\sqrt{m}RL}{\sigma_{\min}^2}$$

with probability at least $1 - \delta/2T$, using that $\sqrt{m}RL/\sigma_{\min} \geq 1$. This, in turn, implies that

$$\|\tilde{\alpha}_t \cdot \vec{e}_t(X_t) - \alpha_t \cdot \vec{e}_t(X_t)\|_{L^2(\rho_t)} \leq \|\tilde{\alpha}_t - \alpha_t\|_2 \|\sqrt{\vec{e}_t(X_t) \cdot \vec{e}_t(X_t)}\|_{L^2(\rho_t)} \leq \frac{4\epsilon mRL^2}{\sigma_{\min}^2},$$

using that $\|\sqrt{\vec{e}_t(X_t) \cdot \vec{e}_t(X_t)}\|_{L^2(\rho_t)} = \sqrt{\sum_{x \in E} \rho_t(x) \sum_{k=1}^m e_{t,k}^2(x)} \leq \sqrt{m}L$. Next, we bound

$$\begin{aligned} & \|\tilde{\alpha}_t \cdot \vec{e}_t(X_t) - \mathbb{E}[Z_{\tilde{\tau}_{t+1}} | X_t]\|_{L^2(\rho_t)} \\ & \leq \|\tilde{\alpha}_t \cdot \vec{e}_t(X_t) - \alpha_t \cdot \vec{e}_t(X_t)\|_{L^2(\rho_t)} + \|\alpha_t \cdot \vec{e}_t(X_t) - \mathbb{E}[Z_{\tilde{\tau}_{t+1}} | X_t]\|_{L^2(\rho_t)} \\ & \leq \frac{4\epsilon mRL^2}{\sigma_{\min}^2} + \min_{a \in \mathbb{R}^m} \|a \cdot \vec{e}_t(X_t) - \mathbb{E}[Z_{\tilde{\tau}_{t+1}} | X_t]\|_{L^2(\rho_t)}, \end{aligned} \quad (12)$$

using that $\alpha_t = \arg \min_{a \in \mathbb{R}^m} \mathbb{E}[(Z_{\tilde{\tau}_{t+1}} - a \cdot \vec{e}_t(X_t))^2]$ minimizes the least square estimator.

Finally, by the union bound, with probability at least $1 - \delta$, Eq. (12) holds for all $t \in [T - 1]$, together with $|\tilde{Z}_{\tilde{\tau}_1} - \mathbb{E}[Z_{\tilde{\tau}_1}]| \leq \epsilon$. Lemma 17 then leads to

$$|\tilde{Z}_{\tilde{\tau}_1} - \mathbb{E}[Z_{\tilde{\tau}_1}]| \leq \frac{8T\epsilon mRL^2}{\sigma_{\min}^2} + 2 \sum_{t=1}^{T-1} \min_{a \in \mathbb{R}^m} \|a \cdot \vec{e}_t(X_t) - \mathbb{E}[Z_{\tilde{\tau}_{t+1}} | X_t]\|_{L^2(\rho_t)},$$

which implies Eq. (11) by using $|\max\{a_0, a_1\} - \max\{a_0, a_2\}| \leq |a_1 - a_2|$ with $a_0, a_1, a_2 \in \mathbb{R}$ on the definition of $\tilde{\mathcal{U}}_0$ in Eq. (4). \blacktriangleleft

Note that the approximation errors $\min_{a \in \mathbb{R}^m} \|a \cdot \vec{e}_t(X_t) - \mathbb{E}[Z_{\tilde{\tau}_{t+1}} | X_t]\|_{L^2(\rho_t)}$ appearing in Theorem 19 depend on the approximated stopping times $\tilde{\tau}_{t+1}$, which in turn depend on $\mathcal{H}_{t'}$ for $t' > t$. It is possible to restate Theorem 19 in terms of $\min_{a \in \mathbb{R}^m} \|a \cdot \vec{e}_t(X_t) - \mathbb{E}[Z_{\tau_{t+1}} | X_t]\|_{L^2(\rho_t)}$, which we do in the next theorem by following a similar approach to [93, Theorem 6.1]. The downside is that the time dependence now becomes exponential.

► **Theorem 20.** *Within the setting of Algorithm 2 with input parameters δ and ϵ , let $T\mathcal{T}_{\text{samp}}$ be the sampling cost of the Markov chain and consider a set of linearly independent functions $\{e_{t,k} : E \rightarrow \mathbb{R}\}_{k=1}^m$ for each $t \in [T - 1]$ with $L := \max_{t \in [T-1], k \in [m]} \|e_{t,k}\|_{L^2(\rho_t)}$ and query cost \mathcal{T}_e . Also consider $\{z_t : E \rightarrow \mathbb{R}\}_{t=0}^T$ with $R := \max_{t \in [T]} \|z_t\|_u < \infty$ and query cost \mathcal{T}_z . Moreover, let $\sigma_{\min} := \min_{t \in [T-1]} \sigma_{\min}(A_t) > 0$. Assume that $\sqrt{m}RL/\sigma_{\min} \geq 1$ and define $\mathcal{T}_{\text{total}} := \mathcal{T}_{\text{samp}} + \mathcal{T}_z + \mathcal{T}_e$. Then, for any $\delta \in (0, 1)$ and $\epsilon \in (0, \sigma_{\min}/2]$, Algorithm 2 outputs $\tilde{\mathcal{U}}_0$ such that*

$$\Pr \left[|\tilde{\mathcal{U}}_0 - \mathbb{E}[Z_{\tau_0}]| \geq 5^T \left(\frac{4\epsilon mRL^2}{\sigma_{\min}^2} + \max_{0 < t < T} \min_{a \in \mathbb{R}^m} \|a \cdot \vec{e}_t(X_t) - \mathbb{E}[Z_{\tau_{t+1}} | X_t]\|_{L^2(\rho_t)} \right) \right] \leq \delta$$

in time

$$O \left(\frac{T^2 m^3}{\epsilon} \mathcal{T}_{\text{total}} L(L + R) \log(T) \log(Tm^2/\delta) \log^{3/2}(mL(L + R)/\epsilon) \log \log(mL(L + R)/\epsilon) \right).$$

Proof. The proof follows the same steps of the proof of Theorem 19, with the further observation in Eq. (12) that if

$$\|\tilde{\alpha}_\ell \cdot \vec{e}_\ell(X_\ell) - \mathbb{E}[Z_{\tilde{\tau}_{\ell+1}} | X_\ell]\|_{L^2(\rho_\ell)} \leq \epsilon_0 + \min_{a \in \mathbb{R}^m} \|a \cdot \vec{e}_\ell(X_\ell) - \mathbb{E}[Z_{\tilde{\tau}_{\ell+1}} | X_\ell]\|_{L^2(\rho_\ell)}, \quad (13)$$

for all $\ell \in \{t, \dots, T-1\}$, where we defined $\epsilon_0 := \frac{4\epsilon m R L^2}{\sigma_{\min}^2}$, then

$$2(T-\ell)\epsilon_0 + 2 \sum_{k=\ell}^{T-1} \min_{a \in \mathbb{R}^m} \|a \cdot \vec{e}_k(X_k) - \mathbb{E}[Z_{\tilde{\tau}_{k+1}} | X_k]\|_{L^2(\rho_k)} \leq 5^{T-\ell}(\epsilon_0 + M_\ell^*), \quad (14)$$

for all $\ell \in \{t, \dots, T-1\}$, where $M_\ell^* := \max_{k=\ell, \dots, T-1} (\min_{a \in \mathbb{R}^m} \|a \cdot \vec{e}_k(X_k) - \mathbb{E}[Z_{\tilde{\tau}_{k+1}} | X_k]\|_{L^2(\rho_k)})$. We prove this bound using backward induction as follows. Eq. (14) clearly holds for $\ell = T-1$. Assume it holds for $\ell = t+1$, we shall prove it is also true for $\ell = t$. First notice that, by the triangle inequality followed by Lemma 17 and then Eq. (13),

$$\begin{aligned} & \min_{a \in \mathbb{R}^m} \|a \cdot \vec{e}_t(X_t) - \mathbb{E}[Z_{\tilde{\tau}_{t+1}} | X_t]\|_{L^2(\rho_t)} \\ & \leq \min_{a \in \mathbb{R}^m} \|a \cdot \vec{e}_t(X_t) - \mathbb{E}[Z_{\tau_{t+1}} | X_t]\|_{L^2(\rho_t)} + \|\mathbb{E}[Z_{\tau_{t+1}} | X_t] - \mathbb{E}[Z_{\tilde{\tau}_{t+1}} | X_t]\|_{L^2(\rho_t)} \\ & \leq \min_{a \in \mathbb{R}^m} \|a \cdot \vec{e}_t(X_t) - \mathbb{E}[Z_{\tau_{t+1}} | X_t]\|_{L^2(\rho_t)} + 2 \sum_{k=t+1}^{T-1} \|\tilde{\alpha}_k \cdot \vec{e}_k(X_k) - \mathbb{E}[Z_{\tilde{\tau}_{k+1}} | X_k]\|_{L^2(\rho_k)} \\ & \leq \min_{a \in \mathbb{R}^m} \|a \cdot \vec{e}_t(X_t) - \mathbb{E}[Z_{\tau_{t+1}} | X_t]\|_{L^2(\rho_t)} + 2 \sum_{k=t+1}^{T-1} (\epsilon_0 + \min_{a \in \mathbb{R}^m} \|a \cdot \vec{e}_k(X_k) - \mathbb{E}[Z_{\tilde{\tau}_{k+1}} | X_k]\|_{L^2(\rho_k)}) \\ & \leq M_t^* + 2(T-t-1)\epsilon_0 + 2 \sum_{k=t+1}^{T-1} \min_{a \in \mathbb{R}^m} \|a \cdot \vec{e}_k(X_k) - \mathbb{E}[Z_{\tilde{\tau}_{k+1}} | X_k]\|_{L^2(\rho_k)}. \end{aligned}$$

Using the above inequality followed by the induction hypothesis,

$$\begin{aligned} & 2(T-t)\epsilon_0 + 2 \sum_{k=t}^{T-1} \min_{a \in \mathbb{R}^m} \|a \cdot \vec{e}_k(X_k) - \mathbb{E}[Z_{\tilde{\tau}_{k+1}} | X_k]\|_{L^2(\rho_k)} \\ & \leq 2\epsilon_0 + 2M_t^* + 6(T-t-1)\epsilon_0 + 6 \sum_{k=t+1}^{T-1} \min_{a \in \mathbb{R}^m} \|a \cdot \vec{e}_k(X_k) - \mathbb{E}[Z_{\tilde{\tau}_{k+1}} | X_k]\|_{L^2(\rho_k)} \\ & \leq 2\epsilon_0 + 2M_t^* + 3 \cdot 5^{T-t-1}(\epsilon_0 + M_{t+1}^*) \\ & \leq 5^{T-t}(\epsilon_0 + M_t^*), \end{aligned}$$

proving the induction statement. The theorem follows by taking $\ell = 0$ in Eq. (14) and using $|\max\{a_0, a_1\} - \max\{a_0, a_2\}| \leq |a_1 - a_2|$, $a_0, a_1, a_2 \in \mathbb{R}$, on the definition of $\tilde{\mathcal{U}}_0$ in Eq. (4). ◀

Given the above theorems, we prove a classical analogue.

► **Theorem 21.** *Within the setting of Algorithm 1, consider N independent sample paths with sample cost $\mathcal{T}_{\text{samp}}$ and let the linearly independent functions $\{e_{t,k} : E \rightarrow \mathbb{R}\}_{k=1}^m$ for each $t \in [T-1]$ be such that $L := \max_{t \in [T-1], k \in [m]} \|e_{t,k}\|_{L^2(\rho_t)}$ and have query cost \mathcal{T}_e . Also consider $\{z_t : E \rightarrow \mathbb{R}\}_{t=0}^T$ with $R := \max_{t \in [T]} \|z_t\|_u < \infty$ and query cost \mathcal{T}_z . Moreover, let $\sigma_{\min} := \min_{t \in [T-1]} \sigma_{\min}(A_t) > 0$ and assume that $\sqrt{m}RL/\sigma_{\min} \geq 1$. Then, for any $\epsilon \in (0, \sigma_{\min}/2]$, Algorithm 1 runs for $O(Tm^2N + Tm^{\omega_*} + TN(\mathcal{T}_{\text{samp}} + \mathcal{T}_z + m\mathcal{T}_e))$ time and returns an estimate $\tilde{\mathcal{U}}_0$ such that*

$$\Pr \left[\|\tilde{\mathcal{U}}_0 - \mathbb{E}[Z_{\tau_0}]\| \geq 5^T \left(\frac{4\epsilon m R L^2}{\sigma_{\min}^2} + \max_{0 < t < T} \min_{a \in \mathbb{R}^m} \|a \cdot \vec{e}_t(X_t) - \mathbb{E}[Z_{\tau_{t+1}} | X_t]\|_{L^2(\rho_t)} \right) \right] \leq 6m^2 e^{-2N\epsilon^2/m^2}.$$

Proof. The error analysis is very similar to that of Theorems 19 and 20, therefore we shall just point out the required changes. Each entry of A_t and b_t is approximated using a Chernoff bound, i.e., $\Pr[|(A_t)_{jl} - (\tilde{A}_t)_{jl}| \geq \epsilon/m] \leq 2e^{-2N\epsilon^2/m^2}$ and $\Pr[|(b_t)_j - (\tilde{b}_t)_j| \geq \epsilon/\sqrt{m}] \leq 2e^{-2N\epsilon^2/m}$ for all $j, l \in [m]$. Moreover, $\Pr[|\tilde{Z}_{\tau_1} - \mathbb{E}[Z_{\tau_1}]| \geq \epsilon] \leq 2e^{-2N\epsilon^2}$. Therefore, by the union bound, all approximations hold with probability at least $1 - 2m^2e^{-2N\epsilon^2/m^2} - 2me^{-2N\epsilon^2/m} - 2e^{-2N\epsilon^2} \geq 1 - 6m^2e^{-2N\epsilon^2/m^2}$.

Regarding the time complexity, the most expensive computational steps are calculating the matrices \tilde{A}_t , which requires $O(Tm^2N)$ time, and inverting them and computing the vectors $\tilde{\alpha}_t$, which requires $O(Tm^{\omega_*})$ time. Sampling $(X_t^{(1)}, \dots, X_t^{(N)})_{t=0}^T$ and querying $(Z_t^{(1)}, \dots, Z_t^{(N)})_{t=0}^T$ and $(e_{t,k}(X_t^{(1)}), \dots, e_{t,k}(X_t^{(N)}))_{t \in [T-1], k \in [m]}$ require $O(NT(\mathcal{T}_{\text{samp}} + \mathcal{T}_z + m\mathcal{T}_e))$ time. All the other steps, computing $\frac{1}{N} \sum_{n=1}^N Z_{\tau_{t+1}}^{(n)} \vec{e}(X_t^{(n)})$, $\tilde{\mathcal{U}}_0$ and $\tilde{\alpha}_t \cdot \vec{e}_t(X_t^{(n)})$ require $O(TmN)$ or $O(Tm)$ time. \blacktriangleleft

If $\mathcal{T}_{\text{samp}}, \mathcal{T}_z, \mathcal{T}_e = O(1)$, then the complexity is $O(Tm^2N + Tm^{\omega_*})$. The factor Tm^2 comes from computing $\{A_t\}_{t=1}^{T-1}$ and accounts for runtime instead of only number of samples.

We summarize and compare the results from Theorems 21 and 20 into a single corollary.

► **Corollary 22.** *Within the setting of Algorithm 1 with input parameters δ, N and Algorithm 2 with input parameters δ, ϵ_0 , let $T\mathcal{T}_{\text{samp}}$ be the sampling cost of the Markov chain and consider a set of linearly independent functions $\{e_{t,k} : E \rightarrow \mathbb{R}\}_{k=1}^m$ for $t \in [T-1]$ with query cost \mathcal{T}_e and $L := \max_{t \in [T-1], k \in [m]} \|e_{t,k}\|_{L^2(\rho_t)}$. Also consider $\{z_t : E \rightarrow \mathbb{R}\}_{t=0}^T$ with query cost \mathcal{T}_z and $R := \max_{t \in [T]} \|z_t\|_u < \infty$. Let $\sigma_{\min} \leq \min_{t \in [T-1]} \sigma_{\min}(A_t)$ be known. Assume that $\sqrt{mRL}/\sigma_{\min} \geq 1$ and define $\mathcal{T}_{\text{total}} := \mathcal{T}_{\text{samp}} + \mathcal{T}_z + \mathcal{T}_e$. For any $\delta \in (0, 1)$ and $\epsilon \in (0, \frac{2mRL^2}{\sigma_{\min}}]$, if $\epsilon_0 = \frac{\epsilon\sigma_{\min}^2}{4mRL^2}$ and $N = \lceil \frac{m^2}{2\epsilon_0^2} \log(6m^2/\delta) \rceil$, then Algorithms 1 and 2 output $\tilde{\mathcal{U}}_0$ such that*

$$\Pr \left[|\tilde{\mathcal{U}}_0 - \mathbb{E}[Z_{\tau_0}]| \geq 5^T \left(\epsilon + \max_{0 < t < T} \min_{a \in \mathbb{R}^m} \|a \cdot \vec{e}_t(X_t) - \mathbb{E}[Z_{\tau_{t+1}} | X_t]\|_{L^2(\rho_t)} \right) \right] \leq \delta$$

using time, respectively,

$$O \left(\frac{Tm^6 R^2 L^4}{\epsilon^2 \sigma_{\min}^4} \mathcal{T}_{\text{total}} \log(m^2/\delta) \right)$$

and (up to log log factors)

$$O \left(\frac{T^2 m^4 RL^3(L+R)}{\epsilon \sigma_{\min}^2} \mathcal{T}_{\text{total}} \log(T) \log \left(\frac{Tm^2}{\delta} \right) \log^{3/2} \left(\frac{m^2 RL^3(L+R)}{\epsilon \sigma_{\min}^2} \right) \right).$$

Proof. The results concerning the quantum Algorithm 2 were already proven in Theorem 20, we just use that $\epsilon_0 = \frac{\epsilon\sigma_{\min}^2}{4mRL^2}$. It is left to prove the results about the classical Algorithm 1. Indeed, by setting $N = \lceil \frac{m^2}{2\epsilon_0^2} \log(6m^2/\delta) \rceil$ into Theorem 21 with $\epsilon_0 = \frac{\epsilon\sigma_{\min}^2}{4mRL^2}$, we obtain

$$\Pr \left[|\tilde{\mathcal{U}}_0 - \mathbb{E}[Z_{\tau_0}]| \geq 5^T \left(\epsilon + \max_{0 < t < T} \min_{a \in \mathbb{R}^m} \|a \cdot \vec{e}_t(X_t) - \mathbb{E}[Z_{\tau_{t+1}} | X_t]\|_{L^2(\rho_t)} \right) \right] \leq \delta.$$

The final time complexity becomes then $O(Tm^2N + Tm^{\omega_*} + TN(\mathcal{T}_{\text{samp}} + \mathcal{T}_z + m\mathcal{T}_e)) = O \left(\frac{Tm^4}{\epsilon^2} (m^2 + \mathcal{T}_{\text{samp}} + \mathcal{T}_z + m\mathcal{T}_e) \frac{R^2 L^4}{\sigma_{\min}^4} \log \left(\frac{m^2}{\delta} \right) \right) = O \left(\frac{Tm^6 R^2 L^4}{\epsilon^2 \sigma_{\min}^4} \mathcal{T}_{\text{total}} \log \left(\frac{m^2}{\delta} \right) \right)$. \blacktriangleleft

► **Remark 23.** We note that in the previous theorem it is necessary to know a lower bound on $\min_{t \in [T-1]} \sigma_{\min}(A_t)$ since it must be inputted into quantum subroutines for Monte Carlo to bound the variance. It is possible, though, to get around this detail by using the more complicated algorithm from [41] which does not require a bound on the variance.

The Parametrized Complexity of Quantum Verification

Srinivasan Arunachalam  

IBM Quantum, Thomas J Watson Research Center, Yorktown Heights, NY, USA

Sergey Bravyi   

IBM Quantum, Thomas J Watson Research Center, Yorktown Heights, NY, USA

Chinmay Nirkhe   

IBM Quantum, Thomas J Watson Research Center, Yorktown Heights, NY, USA

Electrical Engineering and Computer Sciences, University of California, Berkeley, CA, USA

Challenge Institute for Quantum Computation, University of California, Berkeley, CA, USA

Bryan O’Gorman   

IBM Quantum, Thomas J Watson Research Center, Yorktown Heights, NY, USA

Abstract

We initiate the study of parameterized complexity of QMA problems in terms of the number of non-Clifford gates in the problem description. We show that for the problem of parameterized quantum circuit satisfiability, there exists a classical algorithm solving the problem with a runtime scaling exponentially in the number of non-Clifford gates but only polynomially with the system size. This result follows from our main result, that for any Clifford + t T -gate quantum circuit satisfiability problem, the search space of optimal witnesses can be reduced to a stabilizer subspace isomorphic to at most t qubits (independent of the system size). Furthermore, we derive new lower bounds on the T -count of circuit satisfiability instances and the T -count of the W -state assuming the classical exponential time hypothesis (ETH). Lastly, we explore the parameterized complexity of the quantum non-identity check problem.

2012 ACM Subject Classification Theory of computation → Quantum computation theory

Keywords and phrases parametrized complexity, quantum verification, QMA

Digital Object Identifier 10.4230/LIPIcs.TQC.2022.3

Related Version *Previous Version*: <https://arxiv.org/abs/2202.08119>

Funding *Sergey Bravyi*: Was supported in part by the IBM Research Frontiers Institute.

Chinmay Nirkhe: Acknowledges support from NSF Quantum Leap Challenges Institute Grant number OMA2016245 and an IBM Quantum PhD internship.

Acknowledgements Part of this work was completed while CN and BO were participants in the Simons Institute for the Theory of Computing *Summer Cluster on Quantum Computation*. Additionally, we thank Sam Gunn, Zeph Landau, Dimitri Maslov and Kristan Temme for insightful discussions.

1 Introduction

The solutions to many important computational problems require resources that seem to scale exponentially in the system size in the worst case. Parameterized complexity refines this phenomenon by identifying one or more parameters along with algorithms that scale exponentially only in the identified parameters (but polynomially in system size). In the worst case, these parameters typically scale with system size, but often interesting instances have an intermediate value of the parameter. In quantum computing, parameterized complexity has



© Srinivasan Arunachalam, Sergey Bravyi, Chinmay Nirkhe, and Bryan O’Gorman; licensed under Creative Commons License CC-BY 4.0

17th Conference on the Theory of Quantum Computation, Communication and Cryptography (TQC 2022).

Editors: François Le Gall and Tomoyuki Morimae; Article No. 3; pp. 3:1–3:18

Leibniz International Proceedings in Informatics



LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

been applied to the classical simulation of quantum systems [13, 9, 8, 22, 6, 23]. In particular, parameterizing circuits by the number of non-Clifford gates has yielded many state-of-the-art algorithms for classical simulation. Here, we initiate the study of the parameterized complexity of *non-deterministic* computation, i.e., *quantum verification*. Specifically, we consider QMA (Quantum Merlin-Arthur¹) problems parameterized by the number of non-Clifford gates in their verification circuits and obtain non-trivial upper bounds on the classical complexity of finding an optimal witness and the number of qubits required for its representation.

1.1 The parameterized complexity of quantum circuit satisfiability

The first problem we consider is quantum circuit satisfiability (QCSAT), a canonical QMA-complete problem. In a QCSAT instance, the input is an s -gate quantum circuit U acting on $n + m$ qubits followed by the measurement in the standard basis of any $k > 0$ output qubits.² The goal in the QCSAT problem is to estimate the maximal probability that quantum circuit measurement outputs 1^k when run on input states (i.e., witnesses) of the form $|\psi\rangle \otimes |0^m\rangle$ for $|\psi\rangle \in (\mathbb{C}^2)^{\otimes n}$:

$$\text{Val} \stackrel{\text{def}}{=} \max_{|\psi\rangle \in (\mathbb{C}^2)^{\otimes n}} \langle \psi, 0^m | U^\dagger | 1^k \rangle \langle 1^k | U | \psi, 0^m \rangle. \quad (1)$$

The problem can also be phrased as a promise decision problem in which the goal is to decide if $\text{Val} > a$ (yes instance) or $\text{Val} < b$ (no instance) for $a > b$. The decision problem is known to be QMA-complete when $a = 2/3$ and $b = 1/3$.

To the best of our knowledge, there are no previously known classical algorithms that exploit the structure of the circuit to solve QCSAT in less than $\exp(n)$ time; a simple exponential time algorithm for calculating Val can be achieved by searching over the entire Hilbert space of the witness $|\psi\rangle$. One of the main results of our work is that parameterized QCSAT instances with t T -gates for $t \ll n$ can be solved significantly faster than this naive algorithm. We show that there is a stabilizer subspace isomorphic to $(\mathbb{C}^2)^{\otimes t}$ of the the input Hilbert space which contains all optimal witnesses; here an optimal witness is any input state ψ that maximizes the probability of observing the output 1^k .

► **Theorem 1.** *For every QCSAT instance U with $t \leq n$ T -gates, there is an n -qubit Clifford unitary W and a t -qubit state $|\phi\rangle$ such that $W(|\phi\rangle \otimes |0^{n-t}\rangle)$ is an optimal input state. Furthermore, the Clifford unitary W only depends on the description of U and can be computed in (classical, deterministic) time $\text{poly}(n, s)$.*

This insight can be used to construct a faster algorithm for parameterized QCSAT instances.

► **Theorem 2.** *There exists a classical randomized algorithm that takes as input a parameterized instance of QCSAT problem, a precision parameter $\delta > 0$, and outputs a real random variable ξ such that*

$$0 \leq \xi \leq \text{Val} \quad \text{and} \quad \Pr[\xi \geq (1 - \delta)\text{Val}] \geq \frac{99}{100}. \quad (2)$$

The algorithm has runtime $\text{poly}(n, m, s, t) + O(\delta^{-1}t2^t)$.

¹ The complexity class QMA is the quantum analog of the classical non-deterministic complexity classes, MA and NP [18].

² While one can map the k -qubit measurement to a single measurement, this requires the application of a coherent AND logical gate. Implementing this AND gate requires $\Omega(k)$ non-Clifford gates; in the case of parameterized complexity, this cost may be prohibitive. For this reason, we define the problem in terms of the measurement of multiple qubits in the standard basis.

Quantum circuit satisfiability (QCSAT) is the analog of quantum circuit simulation (QMA- vs. BQP-completeness) in the same way that classical circuit satisfiability is the analog³ of circuit simulation (NP- vs. P-completeness). Recall that it is widely believed that any classical algorithm for parameterized quantum circuit simulation should have a runtime scaling exponentially in t ; the current matching upper-bound scales as $2^{\alpha t}$, where $\alpha < 0.3963$ for exact simulators [23] and $\alpha < 0.23$ for approximate simulators [8]. Our result shows that the QCSAT verification problem is not much harder than its simulation counterpart; the resulting exponential scaling is worse, but there is no exponential dependence on the instance size or n , the size of the witness. This is surprising as *classical* circuit satisfiability is believed to require a runtime scaling exponentially with n (the size of the witness) to solve, while classical circuit simulation is trivially solvable in polynomial-time. Therefore, it would be reasonable to expect that a parameterized QCSAT instance would incur a slowdown scaling exponentially with the witness size n due to non-determinism of the problem and a slowdown scaling exponentially with t due to its quantumness. We instead show that the exponential time scaling can be brought to scale with only t when $t \ll n$. This is the primary power of Theorem 1: to efficiently reduce the search space of optimal inputs from n qubits to t qubits.

Our result may seem surprising in view of the earlier work by Morimae et al. [21] that studied a restricted version of the class QMA where the verifier can only perform Clifford gates. In [21], they found that QMA with a Clifford verifier coincides with the standard QMA. However, the computational model of [21] is different from ours since it allows *adaptive* Clifford gates that can be classically controlled by the outcomes of intermediate measurements. In contrast, here we consider unitary (non-adaptive) verification circuits with all measurements delayed until the end.

Let us briefly sketch the proof of Theorems 1 and 2; complete proofs are provided in Section 2. By definition, solving QCSAT is equivalent to estimating the largest eigenvalue of an operator $\rho \stackrel{\text{def}}{=} \langle 0^m | U^\dagger | 1^k \rangle \langle 1^k | U | 0^m \rangle$ acting on n qubits. Consider first a simple case when there are no T gates, i.e., $t = 0$ and U is a Clifford circuit. At a high level, ρ describes a state (unnormalized) generated by a sequence of Clifford operations: (1) initializing each qubit in a basis state or a maximally mixed state, (2) applying a unitary Clifford gate, and (3) post-selectively measuring a qubit in the standard basis. Such operations are known to have very limited computational power – they always produce a (mixed) stabilizer-type state [14]. Accordingly, the largest eigenvalue of ρ can be efficiently computed using the standard stabilizer formalism [10]. Suppose now that U contains t T -gates. It is well-known [5] that a T -gate can be implemented by a gadget that includes only Clifford operations and consumes one copy of a magic state $|A\rangle \propto |0\rangle + e^{i\pi/4}|1\rangle$. Replacing each T -gate in U by this gadget one gets $\rho = \text{Tr}_{\{[t]\}}(|A^t\rangle\langle A^t| \otimes \mathbb{I}_n \rho')$, where ρ' is a bipartite stabilizer state of $t + n$ qubits and we trace out the first t qubits. Our key technical tool is the characterization of bipartite stabilizer states [7]. This result implies that $\rho' = (C_1^\dagger \otimes C_2^\dagger) \rho'' (C_1 \otimes C_2)$, where C_i are unitary Clifford operators and ρ'' is a tensor product of local single-qubit stabilizer states and at most t two-qubit stabilizer states shared between the two subsystems. Using this decomposition we are able to show that $\rho = C_2^\dagger (\rho_{\text{hard}} \otimes \rho_{\text{easy}}) C_2$, where ρ_{hard} is some (non-stabilizer) state of t qubits and ρ_{easy} is a stabilizer state of $n - t$ qubits whose eigenvalues are easy to compute. Thus, QCSAT reduces to estimating the largest eigenvalue of the t -qubit state ρ_{hard} . We remark that this theorem also holds if the T -gate is replaced by an arbitrary angle Z -rotation (since we use a post-selective magic state injection gadget, in which case

³ Technically, they are the analogs of randomized circuit satisfiability and randomized circuit simulation (MA- vs. BPP-completeness).

3:4 The Parametrized Complexity of Quantum Verification

it does not matter if the rotation angle is $\pi/8$ or not). To prove Theorem 2 we make use of the special structure of the state ρ_{hard} and reduce the problem of computing its largest eigenvalue to computing the largest Schmidt coefficient of a certain pure bipartite state of at most t qubits. The latter is computed using the power method with a random starting state [19].

Implications for QCMA vs. QMA

The description of a circuit generating a quantum state can constitute a classical witness for that state. It is an open question (QCMA vs. QMA) in complexity theory if all quantum witnesses have efficient classical descriptions [3, 2, 11]. Our work makes progress on the parameterized version of the question by proving that the witness to any QCSAT instance with t T -gates can be constructed with at most $\exp(t)$ T -gates as it only requires t qubits to describe. An interesting open question is if our techniques lend themselves to any sub-exponential in t upper-bound on the T -count of optimal witness states.

1.2 Implied lower bounds from the exponential time hypothesis

Theorem 2 provides an upper bound on the runtime of a classical algorithm for QCSAT with respect to the number of T -gates in the verifier. In conjunction with other complexity-theoretic assumptions, this also implies a *lower bound* on the T -count of the verification circuit. One such assumption is the Exponential-Time Hypothesis (ETH), introduced by Impagliazzo and Paturi [15]. Informally, ETH is the conjecture that classical k -SAT requires exponential classical time. By Theorem 2, a verifier circuit for QMA with $o(n)$ T -gates would imply a $2^{o(n)}$ -time algorithm for k -SAT, because $\text{NP} \subset \text{QMA}$. Thus we get the following lower bound.

► **Corollary 3.** *ETH implies that any QMA-complete family of Clifford+ T verifier circuits must include circuits with $\Omega(n)$ T -gates, where n is the size of the witness.*

Interestingly, we can also use ETH and Theorem 2 to get a lower bound on the T -count of a quantum circuit that prepares the m -qubit W -state

$$|W_m\rangle = \frac{1}{\sqrt{m}} (|100 \cdots 0\rangle + |010 \cdots 0\rangle + \cdots + |000 \cdots 1\rangle). \quad (3)$$

► **Corollary 4.** *ETH implies that any Clifford+ T circuit V that, when applied to the all-zero state, outputs $|W_m\rangle \otimes |\text{junk}\rangle$ must include $\Omega(m)$ T gates.*

Proofs of both corollaries are provided in Section 3. To our knowledge, this is the first lower bound for the T -count of state preparation based on complexity-theoretic assumptions. The proof uses the NP-hardness [4] of Hamiltonians of the form $H = \sum_{\{i,j\} \in E} Z_i Z_j + \sum_{i \in V} Z_i$. We show that a verifier circuit can be built from only a W state and Clifford operations.

1.3 The complexity of the non-identity check problem

The second QMA-complete problem we consider in this work is the Non-Identity Check (NIC) problem: given a classical description of an n -qubit quantum circuit U , decide if U is close to the identity operation, i.e., is $\min_{\phi} \|U - e^{i\phi} \cdot \mathbb{I}\| \geq c$ or $\leq s$ for $c - s \geq 1/\text{poly}(n)$, promised

one of them is the case.⁴ NIC was first considered by Janzing et al. [16] who showed that this problem is QMA-complete, by reducing it to the QCSAT problem. Subsequently, Ji and Wu [17] showed that the NIC problem remains QMA-complete for even depth-2 circuits with *arbitrary* gates. This motivates a natural question: what is the parameterized complexity of NIC? In particular, how does the complexity of NIC scale with the number of non-Clifford gates? Below we show that NIC can be solved in polynomial-time for Clifford circuits.

► **Theorem 5.** *NIC for Clifford circuits is contained in P.*

We sketch the proof of this theorem here while a complete proof is provided in Section 4. By definition, solving NIC for a Clifford circuit U on n qubits is equivalent to estimating the maximum eigenvalue of a Hamiltonian $H = (\mathbb{I} - U)^\dagger(\mathbb{I} - U)$, that is, checking if $\|H\| \geq c^2$ or $\leq s^2$. In the former case, observe that $\text{Tr}(H^p) \geq c^{2p}$ and in the latter case $\text{Tr}(H^p) \leq 2^n \cdot s^{2p}$. Furthermore, since $c - s \geq 1/\text{poly}(n)$, we can pick $p = \text{poly}(n)$ to make $c^{2p} \gg 2^n \cdot s^{2p}$. Thus it suffices to estimate $\text{Tr}(H^p)$ with $p = \text{poly}(n)$. To this end, observe that H^p can be written as a weighted sum of Clifford powers U^i with $i \in \{-p, \dots, p\}$. Thus $\text{Tr}(H^p) = \sum_{i=-p}^p \alpha_i \text{Tr}(U^i)$ and the coefficients α_i can be efficiently computed using a simple recursive formula. Furthermore, we can efficiently compute $\text{Tr}(U^i)$ for every i using the identity $\text{Tr}(U^i) = 2^n \langle \Phi^{\otimes n} | U^i \otimes \mathbb{I} | \Phi^{\otimes n} \rangle$, where $|\Phi\rangle$ is the EPR state. The right-hand side is the inner product between two stabilizer states of $2n$ qubits. Such inner products can be computed exactly in time $O(n^3)$, see [12, 8]. Putting all this together, we can compute $\text{Tr}(H^p)$ exactly in time $\text{poly}(n)$, which determines if this is a “yes” or “no” instance of the NIC problem.

In addition, we also prove that the NIC problem for constant-depth circuits using gates from a constant-sized gate set is solvable for vanishing completeness parameters. This is in contrast to [17] which shows constant-depth circuits using gates from a general gate set is QMA-complete.

► **Theorem 6.** *Let \mathcal{G} be any constant-sized gate set of 1 and 2 qudit gates and U a quantum circuit of depth at most $t = O(1)$ acting on n qudits of fixed finite local dimension d . Let $a < b$ be any parameters such that $a(n) = o(1)$. Then the NIC problem (a, b) for this restricted class of circuits is in P.*

Our success at understanding parameterized QCSAT came from our characterization of optimal witness states of parameterized verifier circuits as small linear combinations of stabilizers. However, the eigenvectors of parameterized NIC circuits U do not have such a characterization. We need to develop new tools to classically describe the eigenvectors of U in order to achieve an equivalent result for the parameterized NIC problem. Our inability to do so might suggest that the parameterized problem is harder than we previously suspected.

We conclude with the following intriguing question regarding the parameterized complexity of NIC problems. Since the algorithm for NIC for Clifford circuits breaks down in the presence of a single non-Clifford gate, the question of parameterized complexity of NIC is left largely open. It may happen that this problem becomes hard (e.g., NP-hard) in the presence of a single non-Clifford gate.

⁴ We remark that the operator norm is important here; the problem is in BQP if the goal is to decide if $\|U - \mathbb{I}\|_2$ is small or large, where $\|\cdot\|_2$ is the normalized-2 norm [20].

2 Proofs of Theorems 1 and 2

► **Theorem 1.** *For every QCSAT instance U with $t \leq n$ T -gates, there is an n -qubit Clifford unitary W and a t -qubit state $|\phi\rangle$ such that $W(|\phi\rangle \otimes |0^{n-t}\rangle)$ is an optimal input state. Furthermore, the Clifford unitary W only depends on the description of U and can be computed in (classical, deterministic) time $\text{poly}(n, s)$.*

Proof. Suppose U is a Clifford+ T circuit with $c = s - t$ Clifford gates and t T -gates. We assume that U acts on $n + m$ qubits partitioned into a witness register of n qubits and an ancilla register of m qubits. Let Π_{out} be a projector onto the all-ones state $|1\rangle\langle 1|^{\otimes k}$ applied to some designated output register of k qubits. We assume that Π_{out} acts trivially on the remaining $n + m - k$ qubits. Define the *maximum acceptance probability* of U as

$$\pi(U) = \max_{\psi} \langle \psi \otimes 0^m | U^\dagger \Pi_{\text{out}} U | \psi \otimes 0^m \rangle = \max_{\psi} \|\Pi_{\text{out}} U | \psi \otimes 0^m \rangle\|^2, \quad (4)$$

where the maximum is over all normalized n -qubit witness states $|\psi\rangle$. Let

$$\pi(U, \psi) := \|\Pi_{\text{out}} U | \psi \otimes 0^m \rangle\|^2. \quad (5)$$

We shall implement each T -gate in U by the following well-known post-selection gadget:

$$\boxed{T} = \begin{array}{c} \text{---} \bullet \text{---} \\ | \\ \oplus \\ \text{---} \diagdown \text{---} \\ \text{---} \diagup \text{---} \\ \text{---} \end{array} \quad |0\rangle$$

Here, we measure the output qubit in the standard basis and post-select on a measurement outcome of 0. The gadget consumes one copy of a single-qubit magic state

$$|A\rangle \stackrel{\text{def}}{=} \frac{1}{\sqrt{2}}(|0\rangle + e^{i\pi/4}|1\rangle) \quad (6)$$

and, therefore, we can rewrite $\pi(U, \psi)$ as

$$\pi(U, \psi) = 2^t \|\Pi_{\text{out}}^{(1)} C | \psi \otimes 0^m \otimes A^{\otimes t} \rangle\|^2. \quad (7)$$

Here C is a Clifford circuit acting on $n + m + t$ qubits with $c + t$ gates (with c gates originating from U and t CNOT gates originating from the gadgets) and $\Pi_{\text{out}}^{(1)}$ is a product of Π_{out} and single-qubit projectors $|0\rangle\langle 0|$ applied to the second qubit of each T -gate gadget. The extra factor 2^t takes into account that each gadget succeeds with the probability $1/2$. Define $\Pi_{\text{out}}^{(2)} = C^\dagger \Pi_{\text{out}} C$. Then

$$\pi(U, \psi) = 2^t \|\Pi_{\text{out}}^{(2)} | \psi \otimes 0^m \otimes A^{\otimes t} \rangle\|^2. \quad (8)$$

Let us say that a projector Π acting on n qubits is a *stabilizer projector* if it can be written as $\Pi = C^\dagger (|0^{n-k}\rangle\langle 0^{n-k}| \otimes \mathbb{I}_k) C$ for some integer $k \in [0, n]$ and some unitary Clifford operator C on n qubits. We shall use the following facts.

► **Fact 7** ([14]). *Suppose Π is a stabilizer projector on n qubits. Then $\text{Tr}_{\{t\}}(|0\rangle\langle 0| \otimes \mathbb{I}_{n-1}) \Pi = \sigma \Pi'$ for some $\sigma \in \{0, 1, 1/2\}$ and some stabilizer projector Π' on $n - 1$ qubits. One can compute σ and Π' in time $\text{poly}(n)$.*

► **Fact 8** (Bipartite Stabilizer Projectors [7]). *Suppose Π is a stabilizer projector acting on a bipartite system LR where L and R are arbitrary qubit registers. Then there exist unitary Clifford operators C_L and C_R acting on L and R respectively such that*

$$\Pi = (C_L \otimes C_R)^\dagger \Pi' (C_L \otimes C_R) \quad (9)$$

where Π' is a tensor product of the following one-qubit and two-qubit stabilizer projectors:

- Single-qubit projectors $|0\rangle\langle 0|$ and I .
- Two-qubit projectors $|\Phi^+\rangle\langle\Phi^+|$ where $|\Phi^+\rangle = (|00\rangle + |11\rangle)/\sqrt{2}$.
- Two-qubit projectors $|00\rangle\langle 00| + |11\rangle\langle 11|$.

Furthermore, each two-qubit projector acts on one qubit in L and one qubit in R . The above decomposition can be computed in time $\text{poly}(|L| + |R|)$.

By definition, $\Pi_{\text{out}}^{(2)}$ is a stabilizer projector on $n + m + t$ qubits. Fact 7 implies that

$$\langle 0^m | \Pi_{\text{out}}^{(2)} | 0^m \rangle = \gamma 2^{-r} \Pi_{\text{out}}^{(3)} \quad (10)$$

for some $\gamma \in \{0, 1\}$, integer $r \in \{0, \dots, m\}$, and some stabilizer projector $\Pi_{\text{out}}^{(3)}$ on $n + t$ qubits. From Eqs. (8,10) one gets

$$\pi(U, \psi) = \gamma 2^{t-r} \|\Pi_{\text{out}}^{(3)} |\psi\rangle \otimes A^{\otimes t}\|^2. \quad (11)$$

If $\gamma = 0$ then $\pi(U, \psi) = 0$ for any witness $|\psi\rangle$. Accordingly, one can choose the Clifford unitary W in the statement of the theorem arbitrarily. From now on we assume that $\gamma = 1$. Apply Fact 8 to the stabilizer projector $\Pi_{\text{out}}^{(3)}$ and the partition $[n + t] = LR$ where L is the n -qubit witness register and R is the t -qubit magic state register. We get

$$\Pi_{\text{out}}^{(3)} = (C_L \otimes C_R)^\dagger \Pi_{\text{out}}^{(4)} (C_L \otimes C_R) \quad (12)$$

where $\Pi_{\text{out}}^{(4)}$ is a tensor product of one-qubit and two-qubit stabilizer projectors from Fact 8. Substituting this into Eq. (11) with $\gamma = 1$ one gets

$$\pi(U, \psi) = 2^{t-r} \|\Pi_{\text{out}}^{(4)} C_L |\psi\rangle \otimes C_R |A^{\otimes t}\|^2. \quad (13)$$

Equivalently,

$$\pi(U, C_L^\dagger \psi) = 2^{t-r} \|\Pi_{\text{out}}^{(4)} |\psi\rangle \otimes C_R |A^{\otimes t}\|^2. \quad (14)$$

By definition, the register R contains t qubits. Thus $\Pi_{\text{out}}^{(4)}$ may contain at most t two-qubit projectors $|\Phi^+\rangle\langle\Phi^+|$ and $|00\rangle\langle 00| + |11\rangle\langle 11|$. Indeed, each two-qubit projector must have one qubit in R , see Fact 8. Partition $L = L' L''$ such that each two-qubit projector that appears in $\Pi_{\text{out}}^{(4)}$ has one qubit in L'' and the other qubit in R . Then

$$|L''| \leq t \quad \text{and} \quad \Pi_{\text{out}}^{(4)} = \Gamma_{L'} \otimes \Lambda_{L'' R} \quad (15)$$

for some stabilizer projectors Γ and Λ . Here the subscripts indicate the registers acted upon by each projector. Furthermore, Γ is a tensor product of single-qubit projectors $|0\rangle\langle 0|$ and I . Define an operator

$$\Pi_{\text{out}}^{(5)} \stackrel{\text{def}}{=} {}_R \langle A^{\otimes t} | C_R^\dagger \Lambda_{L'' R} C_R | A^{\otimes t} \rangle_R \quad (16)$$

acting on the register L'' . Note that $\Pi_{\text{out}}^{(5)}$ is Hermitian and positive semi-definite (although $\Pi_{\text{out}}^{(5)}$ might not be a projector). Then

$$\pi(U, C_L^\dagger \psi) = 2^{t-r} \langle \psi | \Gamma_{L'} \otimes \Pi_{\text{out}}^{(5)} | \psi \rangle. \quad (17)$$

Here the tensor product separates L' and L'' . It follows that $|\psi\rangle$ is an optimal witness such that $\pi(U) = \pi(U, \psi)$ if

$$C_L |\psi\rangle = |\phi_{L'}\rangle \otimes |\phi_{L''}\rangle, \quad (18)$$

3:8 The Parametrized Complexity of Quantum Verification

where $|\phi_{L'}\rangle$ is a $(+1)$ -eigenvector of the projector $\Gamma_{L'}$ and $|\phi_{L''}\rangle$ is an eigenvector of $\Pi_{\text{out}}^{(5)}$ with the maximum eigenvalue. From Equation (15) one infers that $|\phi_{L'}\rangle$ is a state of at most t qubits. Since $\Gamma_{L'}$ is a product of $|0\rangle\langle 0|$ and \mathbb{I} terms, divide L' into L'_1 and L'_2 such that

$$\Gamma_{L'} = |0\rangle^{|L'_1|}\langle 0|_{L'_1} \otimes \mathbb{I}_{L'_2}. \quad (19)$$

Then, the minimizing $\phi_{L'}$ has the form $\phi_{L'} = |0\rangle^{|L'_1|}_{L'_1} \otimes |\text{junk}\rangle_{L'_2}$ for any state $|\text{junk}\rangle$. To conclude, one can choose an optimal witness state $|\psi\rangle$ such that

$$|\psi\rangle = C_L^\dagger \left(|0\rangle^{|L'_1|} \otimes |\text{junk}\rangle \otimes |\phi_{L''}\rangle \right) \quad (20)$$

for some $(\leq t)$ -qubit state $|\phi_{L''}\rangle$ and some Clifford operators C_L . This is equivalent to the statement of the theorem. Additionally observe that all the above steps necessary to obtain W can be implemented efficiently since they only involve manipulations with Clifford circuits and stabilizer projectors. \blacktriangleleft

► Theorem 2. *There exists a classical randomized algorithm that takes as input a parametrized instance of QCSAT problem, a precision parameter $\delta > 0$, and outputs a real random variable ξ such that*

$$0 \leq \xi \leq \text{Val} \quad \text{and} \quad \Pr[\xi \geq (1 - \delta)\text{Val}] \geq \frac{99}{100}. \quad (2)$$

The algorithm has runtime $\text{poly}(n, m, s, t) + O(\delta^{-1}t2^t)$.

Proof. First let us introduce some notations. Let $\mathcal{H}(n)$ be the set of all normalized n -qubit pure states. Given a hermitian n -qubit operator M , let $\lambda_{\max}(M) = \max_{\psi \in \mathcal{H}(n)} \langle \psi | M | \psi \rangle$ be the maximum eigenvalue of M . Define two-qubit projectors

$$\Gamma^{(1)} = |\Phi^+\rangle\langle \Phi^+| \quad \text{and} \quad \Gamma^{(2)} = |00\rangle\langle 00| + |11\rangle\langle 11|. \quad (21)$$

Recall that $|\Phi^+\rangle = (|00\rangle + |11\rangle)/\sqrt{2}$. Given a pair of disjoint k -qubit registers R and L , let $\Gamma_{RL}^{(i)}$ be a $(2k)$ -qubit projector that applies $\Gamma^{(i)}$ to the j -th qubit of R and the j -th qubit of L for each $j = 1, \dots, k$. Below we follow notations introduced in the proof of Theorem 1. Our starting point is the expression for the quantity Val derived in Eqs. (16,17,18) thereof, namely

$$\text{Val} = 2^{t-r} \lambda_{\max}(\Pi_{\text{out}}^{(5)}), \quad (22)$$

where $\Pi_{\text{out}}^{(5)}$ is a positive semi-definite operator defined in Eq. (16), namely

$$\Pi_{\text{out}}^{(5)} = {}_R \langle A^{\otimes t} | C_R^\dagger \Lambda_{L''R} C_R | A^{\otimes t} \rangle_R. \quad (23)$$

Recall that R and L'' are disjoint qubit registers such that $|R| = t$, $|L''| \leq t$, C_R is some Clifford operator acting on R , and $\Lambda_{L''R}$ is a product of one- and two-qubit stabilizer projectors from Fact 8 such that each one-qubit projector acts on R and each two-qubit projector acts on both R and L'' . In other words, $\Lambda_{L''R}$ can be written as

$$\Lambda_{L''R} = \Gamma_{L_1R_1}^{(1)} \Gamma_{L_2R_2}^{(2)} |0\rangle\langle 0|_{R_3} \mathbb{I}_{R_4} \quad (24)$$

for some partitions $L'' = L_1L_2$ and $R = R_1R_2R_3R_4$ with $|R_1| = |L_1|$ and $|R_2| = |L_2|$. Below we use notations $r_i = |R_i|$ and $\ell_i = |L_i|$.

We claim that $\Pi_{\text{out}}^{(5)}$ commutes with any Pauli operator Z_j , $j \in L_2$. Indeed, it suffices to check that Z_j commutes with $\Lambda_{L''R}$. The latter acts on the register L_2 by a diagonal operator $\Gamma_{L_2R_2}^{(2)}$ which commutes with Z -type Pauli operators confirming the claim. Thus one can choose the maximum eigenvector ψ of $\Pi_{\text{out}}^{(5)}$ such that $Z_j|\psi\rangle = (-1)^{\sigma_j}|\psi\rangle$ for all $j \in L_2$ and some unknown $\sigma \in \{0,1\}^{\ell_2}$. Equivalently, $|\psi\rangle = |\psi'\rangle_{L_1} \otimes |\sigma\rangle_{L_2} \equiv |\psi'_{L_1} \otimes \sigma_{L_2}\rangle$ for some $\psi' \in \mathcal{H}(\ell_1)$. Using Eq. (22) one gets

$$\text{Val} = 2^{t-r} \max_{\psi \in \mathcal{H}(\ell_1)} \max_{\sigma \in \{0,1\}^{\ell_2}} \langle \psi_{L_1} \otimes \sigma_{L_2} | \Pi_{\text{out}}^{(5)} | \psi_{L_1} \otimes \sigma_{L_2} \rangle. \quad (25)$$

We shall discard the register L_2 using the identity

$${}_{L_2} \langle \sigma | \Gamma_{L_2R_2}^{(2)} | \sigma \rangle_{L_2} = |\sigma\rangle \langle \sigma |_{R_2}. \quad (26)$$

Substituting this identity into Eq. (25) gives

$$\text{Val} = 2^{t-r} \max_{\sigma \in \{0,1\}^{\ell_2}} \lambda_{\max}(\Pi_{\text{out}}^{(6)}(\sigma)) \quad (27)$$

where $\Pi_{\text{out}}^{(6)}(\sigma)$ is a positive semi-definite operator acting on the register L_1 defined as

$$\Pi_{\text{out}}^{(6)}(\sigma) = {}_R \langle A^{\otimes t} | C_R^\dagger \Gamma_{L_1R_1}^{(1)} | \sigma \rangle \langle \sigma |_{R_2} | 0 \rangle \langle 0 |_{R_3} \mathbb{I}_{R_4} C_R | A^{\otimes t} \rangle_R. \quad (28)$$

We shall discard the register L_1 using the quantum teleportation identity

$${}_{L_1} \langle \psi | \Gamma_{L_1R_1}^{(1)} | \psi \rangle_{L_1} = 2^{-\ell_1} |\psi^*\rangle \langle \psi^* |_{R_1} \quad (29)$$

which holds for any state $\psi \in \mathcal{H}(\ell_1)$. Here ψ^* is the complex conjugate of ψ in the standard basis of ℓ_1 qubits. Using the teleportation identity and the definition of $\Pi_{\text{out}}^{(6)}(\sigma)$ one can check that

$$\langle \psi | \Pi_{\text{out}}^{(6)}(\sigma) | \psi \rangle = 2^{-\ell_1} \langle A^{\otimes t} | C_R^\dagger |\psi^*\rangle \langle \psi^* |_{R_1} | \sigma \rangle \langle \sigma |_{R_2} | 0 \rangle \langle 0 |_{R_3} \mathbb{I}_{R_4} C_R | A^{\otimes t} \rangle_R \quad (30)$$

for any state $\psi \in \mathcal{H}(\ell_1)$. At this point both registers L_1 and L_2 have been discarded. After some algebra one can rewrite Eq. (30) as

$$\langle \psi | \Pi_{\text{out}}^{(6)}(\sigma) | \psi \rangle = 2^{-\ell_1} \| {}_{R_1} \langle \psi^* | \varphi(\sigma) \rangle_{R_1R_4} \|^2, \quad (31)$$

where $|\varphi(\sigma)\rangle$ is a state of R_1R_4 defined as

$$|\varphi(\sigma)\rangle = {}_{R_2R_3} \langle \sigma_{R_2}, 0_{R_3} | C_R | A^{\otimes t} \rangle_R. \quad (32)$$

Since the set $\mathcal{H}(\ell_1)$ is closed under the complex conjugation, Eqs. (27,31) give

$$\text{Val} = 2^{t-r-\ell_1} \max_{\sigma \in \{0,1\}^{\ell_2}} \max_{\psi \in \mathcal{H}(\ell_1)} \| {}_{R_1} \langle \psi | \varphi(\sigma) \rangle_{R_1R_4} \|^2. \quad (33)$$

At this point the only remaining registers are R_1 and R_4 . Clearly, the optimal state $\psi \in \mathcal{H}(\ell_1)$ that achieves the maximum in Eq. (33) coincides with the largest eigenvector of a reduced density matrix

$$\rho_{R_1}(\sigma) = \text{Tr}_{R_4} |\varphi(\sigma)\rangle \langle \varphi(\sigma)|. \quad (34)$$

Thus Eq. (33) is equivalent to

$$\text{Val} = 2^{t-r-\ell_1} \max_{\sigma \in \{0,1\}^{\ell_2}} \lambda_{\max}(\rho_{R_1}(\sigma)). \quad (35)$$

3:10 The Parametrized Complexity of Quantum Verification

Recall that any t -qubit Clifford operator can be efficiently compiled to a Clifford circuit with $O(t^2)$ one- and two-qubit gates [1]. Thus one can compute a t -qubit state $|\phi\rangle := C_R|A^{\otimes t}\rangle_R$ as a vector of complex amplitudes in time $\text{poly}(t)2^t$ using the standard state vector simulator of quantum circuits. We assume that ϕ is stored in a classical RAM memory for the rest of the algorithm such that any amplitude of ϕ can be accessed in time $\text{poly}(t)$. By definition, $|\varphi(\sigma)\rangle$ is obtained from $|\phi\rangle$ by projecting the registers R_2R_3 onto the basis state $|\sigma_{R_2}, 0_{R_3}\rangle$, see Eq. (32). Equivalently, $|\varphi(\sigma)\rangle$ is obtained from $|\phi\rangle$ by selecting a subset of $2^{t-r_2-r_3}$ amplitudes. Thus one can compute $|\varphi(\sigma)\rangle$ as a vector of complex amplitudes in time $\text{poly}(t)2^{t-r_2-r_3}$ for any given σ . By definition, $|\varphi(\sigma)\rangle$ is a state of $t - r_2 - r_3$ qubits. We shall use the following fact.

► **Lemma 9.** *Suppose $|\varphi\rangle$ is a pure n -qubit state specified as a vector of complex amplitudes and $\delta > 0$ is a precision parameter. Consider a partition $[n] = AB$, where A and B are disjoint qubit registers. Let $\rho_A = \text{Tr}_B|\varphi\rangle\langle\varphi|$ be the reduced density matrix of A . There exists a classical randomized algorithm that runs in time $O(\delta^{-1}n2^n)$ and outputs a real random variable ξ such that*

$$0 \leq \xi \leq \lambda_{\max}(\rho_A) \quad \text{and} \quad \Pr[\xi \geq (1 - \delta)\lambda_{\max}(\rho_A)] \geq \frac{99}{100}. \quad (36)$$

Proof. Assume wlog that $|A| \leq |B|$ (otherwise switch A and B). We have $\rho_A = \eta\eta^\dagger$, where η is a matrix of size $2^{|A|} \times 2^{|B|}$ with matrix elements $\langle x|\eta|y\rangle = \langle x_A y_B|\varphi\rangle$. Given a list of amplitudes of $|\varphi\rangle$, one can compute the matrix of η in time $O(2^n)$ since $|A| + |B| = n$. For any $|A|$ -qubit state $|v\rangle$ the matrix-vector product $|v\rangle \rightarrow \eta^\dagger|v\rangle$ can be computed in time $O(2^n)$. Likewise, for any $|B|$ -qubit state $|w\rangle$ the matrix-vector product $|w\rangle \rightarrow \eta|w\rangle$ can be computed in time $O(2^n)$. We conclude that the matrix-vector product $|v\rangle \rightarrow \rho_A|v\rangle$ can be computed in time $O(2^n)$.

We shall compute an estimator ξ satisfying Eq. (36) using the power method with a random starting state [19]. Namely, let $|\phi\rangle \in \mathcal{H}(|A|)$ be a Haar-random state of A . Given a number of iterations $q \geq 2$, the power method outputs an estimator

$$\xi_q = \frac{\langle \psi_q | \rho_A | \psi_q \rangle}{\langle \psi_q | \psi_q \rangle}, \quad |\psi_q\rangle := \rho_A^{q-1} |\phi\rangle. \quad (37)$$

Clearly, computing ξ_q requires q matrix-vector multiplications for the matrix ρ_A . Thus the runtime scales as $O(q2^n)$. Note that $0 \leq \xi_q \leq \lambda_{\max}(\rho_A)$ with certainty. Theorem 3.1 of [19] guarantees that the relative error

$$\epsilon_q := \frac{\lambda_{\max}(\rho_A) - \xi_q}{\lambda_{\max}(\rho_A)} \quad (38)$$

obeys

$$\mathbb{E}(\epsilon_q) \leq \frac{0.871 \log(2^{|A|})}{q-1} \leq \frac{n}{q} \quad (39)$$

for all $q \geq 2$. Here the expectation value is taken over the random starting state ϕ and we use the natural logarithm. Since ϵ_q is a non-negative random variable, Markov inequality implies that $\epsilon_q \leq 100 \cdot \mathbb{E}(\epsilon_q)$ with the probability at least 99/100. Thus the desired estimator ξ satisfying Eq. (36) can be chosen as $\xi = \xi_q$ with $q = \lceil 100n/\delta \rceil$. ◀

Applying Lemma 9 to the state $|\varphi(\sigma)\rangle$ of $n = t - r_2 - r_3$ qubits with the registers $A = R_1$ and $B = R_4$ one obtains an estimator $\xi(\sigma)$ satisfying

$$0 \leq \xi(\sigma) \leq \lambda_{\max}(\rho_{R_1}(\sigma)) \quad \text{and} \quad \Pr[\xi(\sigma) \geq (1 - \delta)\lambda_{\max}(\rho_{R_1}(\sigma))] \geq 99/100. \quad (40)$$

The runtime required to compute the estimator $\xi(\sigma)$ for any fixed σ is $O(\delta^{-1}t2^{t-r_2-r_3})$. Thus computing the estimators $\xi(\sigma)$ for all $\sigma \in \{0,1\}^{\ell_2}$ takes time $O(\delta^{-1}t2^t)$. Here we noted that $r_2 = \ell_2$. We choose the desired estimator ξ approximating the quantity Val as

$$\xi = 2^{t-r-\ell_1} \max_{\sigma \in \{0,1\}^{\ell_2}} \xi(\sigma). \quad (41)$$

Let $\sigma^* \in \{0,1\}^{\ell_2}$ be the optimal bit string that achieves the maximum in Eq. (35) such that

$$\text{Val} = 2^{t-r-\ell_1} \lambda_{\max}(\rho_{R_1}(\sigma^*)). \quad (42)$$

From Eq. (40) one infers that $\xi \leq \text{Val}$ with certainty and

$$\Pr[\xi \geq (1-\delta)\text{Val}] \geq \Pr[\xi(\sigma^*) \geq (1-\delta)\lambda_{\max}(\rho_{R_1}(\sigma^*))] \geq 99/100. \quad (43)$$

We conclude by noting that all manipulations performed in the proof of Theorem 1 to compute the Clifford circuit C_R and the stabilizer projector $\Lambda_{L''R}$ can be implemented in time $\text{poly}(n, m, s, t)$ using the standard stabilizer formalism [14]. Thus the total runtime required to compute the desired estimator ξ is

$$\text{poly}(n, m, s, t) + O(\delta^{-1}t2^t). \quad (44)$$

◀

In Appendix A (Theorem 16) we give an alternative algorithm for solving the same problem as Theorem 2 but using slightly different techniques. It has some advantages over Theorem 2 which we elaborate in Appendix A.

3 Implied lower bounds from the Exponential-Time Hypothesis

The Exponential-Time Hypothesis (ETH), introduced by Impagliazzo and Paturi [15], is the conjecture that, informally, (classical) k -SAT requires exponential (classical) time.

► **Definition 10** (Exponential-Time Hypothesis [15]). *Let*

$$s_k \stackrel{\text{def}}{=} \inf \{ \delta : \text{there exists } 2^{\delta n}\text{-time algorithm for solving } k\text{-SAT} \}. \quad (45)$$

Then $s_k > 0$ is a constant for all $k \geq 3$.

It is a stronger assumption than $\text{P} \neq \text{NP}$, which implies just that k -SAT requires superpolynomial-time. We show that ETH, together with Theorem 2, implies T -count lower bounds, which is Corollary 3.

► **Corollary 3.** *ETH implies that any QMA-complete family of Clifford+ T verifier circuits must include circuits with $\Omega(n)$ T -gates, where n is the size of the witness.*

Proof. Consider a family of Clifford + T -gate QMA verifier circuits and let $f(n)$ be the number of T gates in the circuits for n -qubit witnesses. Suppose that $f(n) = o(n)$. Then by Theorem 2, each instance can be solved in $O(\text{poly}(n)2^{o(n)})$ deterministic classical time, which contradicts ETH. The theorem follows from the fact that $\text{NP} \subseteq \text{QMA}$. ◀

Second, we show that Theorem 2 along with ETH implies a linear lower-bound on the T -count complexity of generating a W state. This is Corollary 4.

► **Corollary 4.** ETH implies that any Clifford+T circuit V that, when applied to the all-zero state, outputs $|W_m\rangle \otimes |\text{junk}\rangle$ must include $\Omega(m)$ T gates.

The proof will use the following fact, due to Barahona [4]:

► **Theorem 11** ([4]). Estimating, to within inverse polynomial additive error, the ground state energy of the following class of Hamiltonians is NP-hard:

$$H = \sum_{\{i,j\} \in E} Z_i Z_j + \sum_{i \in V} Z_i, \quad (46)$$

where $G = (V, E)$ is a planar graph with maximum degree 3.

Proof of Corollary 4. Consider a circuit V starting from all-zeros that outputs $|W_m\rangle \otimes |\text{junk}\rangle$ with t T gates. We will show that this implies a $2^{O(t)}$ -time classical algorithm for k -SAT, thus proving the Corollary by contradiction.

Let H' be a diagonal Hamiltonian of the form of Equation (46) with $m' = O(n)$ terms, where n is the number of vertices in the graph. Let $H = H' - m' \leq 0$ be H' shifted by a constant so that it's negative semidefinite. It will be convenient to write H as a sum of $m = 2m'$ terms:

$$H = \sum_{i=1}^m H_i = \sum_{i=1}^{m'} Z_{S_i} - \sum_{i=m'+1}^{2m'} 1 \quad (47)$$

where S_i is a set of one or two indices and $Z_S = \prod_{i \in S} Z_i$.

Let C be the circuit consisting of m gates constructed in the following way. It acts on an m -qubit “control” register A and an n -qubit “computational” register B. For $1 \leq i \leq m'$, each gate C_i implements the i -th term of H (either Z or ZZ) on the corresponding qubits of the computational register, controlled on the i -th qubit of the control register. Because each term of the Hamiltonian is Pauli, the controlled version is Clifford, and so C is a Clifford operator. For $m' < i \leq 2m'$, the corresponding gate is simply Z_i (in the control register).

Let $U \stackrel{\text{def}}{=} (V^\dagger \otimes I) \cdot C \cdot (V \otimes I)$; U has $2t$ T gates. The probability of measuring all zeros on the control register given the input state $U |0^m, \psi\rangle_{A,B}$ is

$$\text{Tr} \left[|0^m\rangle\langle 0^m|_A \otimes I_B V^\dagger C V |0^m, \psi\rangle\langle 0^m, \psi|_{A,B} V^\dagger C^\dagger V \right] \quad (48)$$

$$= \text{Tr} \left[|W^m\rangle\langle W^m|_A \otimes I_B C |W^m, \psi\rangle\langle W^m, \psi|_{A,B} C^\dagger \right] \quad (49)$$

$$= \frac{1}{m^2} \sum_{i,j,k,\ell} \text{Tr} \left[|e_k\rangle\langle e_\ell|_A I_B C |e_i, \psi\rangle\langle e_j, \psi|_{A,B} C^\dagger \right] \quad (50)$$

$$= \frac{1}{m^2} \sum_{i,j,k,\ell} \text{Tr} \left[|e_k\rangle\langle e_\ell|_A I_B H_i |e_i, \psi\rangle\langle e_j, \psi|_{A,B} H_j \right] \quad (51)$$

$$= \frac{1}{m^2} \sum_{i,j} \text{Tr} [H_i |\psi\rangle\langle \psi|_B H_j] \quad (52)$$

$$= \frac{1}{m^2} \sum_{i,j} \langle \psi | H_i H_j | \psi \rangle \langle \psi | H_i H_j | \psi \rangle \quad (53)$$

$$= \frac{1}{m^2} \langle \psi | H^2 | \psi \rangle \langle \psi | H^2 | \psi \rangle. \quad (54)$$

Because H is negative semidefinite, the state ψ that maximizes the probability of measuring all zeros as above also minimizes $\langle \psi | H | \psi \rangle \langle \psi | H | \psi \rangle$. By Theorem 2, such a W -state preparation circuit V with t T-gates implies a $2^{O(t)}$ -time classical algorithm for solving k -SAT. ◀

4 The complexity of quantum non-identity check

► **Definition 12** (Non-Identity Check [16]). *An instance of the non-identity check (NIC) problem is a classical description of a quantum circuit U and two real numbers a, b such that $b > a$ with the promise that*

$$d_{\mathbb{I}}(U) \stackrel{\text{def}}{=} \min_{\phi} \|U - e^{i\phi} \cdot \mathbb{I}\| \quad (55)$$

is either at most a or at least b . The instance is called a yes instance if $d_{\mathbb{I}}(U) \geq b$ and a no instance if $d_{\mathbb{I}}(U) \leq a$.

In this section, we present two scenarios in which the non-identity check problem becomes trivial to solve. It was proved by Ji and Wu [17], that the problem is in general QMA-hard for depth 2 circuits over qudits when $b - a = 1/\text{poly}(n)$ and the quantum gates are specified to $\Omega(\log n)$ bits of precision. We first show that NIC is solvable in P when the entire circuit is Clifford regardless of the depth of the circuit. Second, we show that if $a = o(1)$ is a sub-constant function, then the problem is in P for constant-depth circuits built from a finite gate set.

4.1 Clifford circuits

► **Theorem 5.** *NIC for Clifford circuits is contained in P.*

Proof. In order to see this, let C be the unknown Clifford circuit for which we need to determine whether $\|C - \mathbb{I}\| \geq \alpha$ or $\|C - \mathbb{I}\| \leq \beta$ for $\alpha - \beta \geq 1/\text{poly}(n)$. To this end, consider a Hamiltonian $H = (\mathbb{I} - C)^\dagger(\mathbb{I} - C) = 2\mathbb{I} - C - C^\dagger$, whose norm satisfies $\|H\| = \|\mathbb{I} - C\|^2 \leq 4$. In order to solve NIC(C), we need to decide if $\|H\| \geq \alpha^2$ or $\|H\| \leq \beta^2$. In the former case observe that $\text{Tr}(H^p) \geq \alpha^{2p}$ and in the latter case we have $\text{Tr}(H^p) \leq 2^n \cdot \beta^{2p}$. Since $\alpha - \beta \geq 1/\text{poly}(n)$, it suffices to pick $p = \text{poly}(n)$, in order to satisfy $\alpha^{2p} \geq 2 \cdot 2^n \cdot \beta^{2p}$. Hence if an algorithm could estimate $\text{Tr}(H^p)$ well enough, then it can distinguish if C satisfies the Yes or No instance of NIC problem.

We now show how to compute $\text{Tr}(H^p)$ for $p = \text{poly}(n)$ exactly and efficiently. In this direction, observe that we can express H^p in terms of sum of Clifford powers, i.e.,

$$\text{Tr}(H^p) = \text{Tr} \left(\sum_{i=-p}^p a_i C^i \right) = \sum_{i=-p}^p a_i \text{Tr}(C^i) \quad (56)$$

for some coefficients $a_i \in \mathbb{R}$. Furthermore, since H assumes a simple form $H = \mathbb{I} - C$, the $(2p + 1)$ coefficients a_i can be computed explicitly in $\text{poly}(n)$ time. Now, it remains to compute $\text{Tr}(C^i)$ for each one of the $2p + 1$ terms. The trace of a Clifford power can be computed exactly by observing that $\text{Tr}(C^i) = 2^n \langle \Phi | \mathbb{I} \otimes C^i | \Phi \rangle$ where $|\Phi\rangle = \frac{1}{\sqrt{2^n}} \sum_i |i, i\rangle$ is the maximally entangled state on $(2n)$ -qubits. Observe that since $|\Phi\rangle$ is a stabilizer state, we have that $|\psi\rangle = \mathbb{I} \otimes C^i |\Phi\rangle$ is also a $(2n)$ -qubit stabilizer state. It remains to estimate inner product between two stabilizer states $|\Phi\rangle$ and $|\psi\rangle$. It is known that the inner product between any n -qubit stabilizer states can be computed exactly (including the overall phase) in time $O(n^3)$, see [12, 8]. Therefore, one can compute each one of the $\text{Tr}(C^i)$ and a_i in Equation (56) in time $\text{poly}(n)$ and overall since $p = \text{poly}(n)$, we can exactly compute $\text{Tr}(H^p)$ in time $\text{poly}(n)$. This suffices to decide if $\text{Tr}(H^p) \geq \alpha^{2p}$ (the YES instance of NIC) or $\text{Tr}(H^p) \leq 2^n \cdot \beta^{2p}$ (the NO instance of NIC) for $\alpha - \beta \geq 1/\text{poly}(n)$. ◀

4.2 Constant-sized gate sets

In this section, we are going to show that the NIC problem for $a = o(1)$, is in P if we restrict ourselves to circuits of constant depth and a constant-sized gate set. Curiously, the problem was shown to be QMA-hard when either we use an arbitrary gate set or constant-sized gate sets, but we allow circuits of $\Omega(\log n)$ -depth [17].

► **Theorem 6.** *Let \mathcal{G} be any constant-sized gate set of 1 and 2 qudit gates and U a quantum circuit of depth at most $t = O(1)$ acting on n qudits of fixed finite local dimension d . Let $a < b$ be any parameters such that $a(n) = o(1)$. Then the NIC problem (a, b) for this restricted class of circuits is in P.*

For this proof, we will need a few definitions and preliminary lemmas which we list here first and prove after the proof of the theorem. First, we will need a wonderful fact about low-depth circuits that the reduced density matrix $\text{tr}_{-i} U\psi U^\dagger$ only depends on the lightcone of the i th qudit.

► **Fact 13.** *Consider a quantum state ψ on n qudits and U a quantum circuit. For any subset A of the qudits, let L_A be the support of the lightcone of A with respect to U . Then,*

$$\text{tr}_{-A}(U\psi U^\dagger) = \text{tr}_{-A} \left(U_{L_A} (\psi_{L_A} \otimes \nu_{-L_A}) U_{L_A}^\dagger \right) \quad (57)$$

where ν is the maximally mixed quantum state and U_{L_A} the circuit restricted to gates contained in L_A .

Second, we notice that if a quantum circuit U is close to identity overall, then the reduced action of the circuit on any region must also be close to identity. We will often use the contrapositive of this statement: if the reduced action of a circuit on any small region is far from identity, then the circuit overall is far from identity.

► **Fact 14.** *Let U be a quantum circuit on n qudits and let a be a constant such that $d_{\mathbb{I}}(U) < a$. Then for all states ψ and all regions A ,*

$$\left\| \text{tr}_{-A}(U\psi U^\dagger) - \psi_A \right\| \leq a. \quad (58)$$

Proof of Theorem 6. Let $\mathcal{C}(\ell, h)$ be the collection of all quantum circuits acting on ℓ qudits and of depth $\leq h$ consisting of gates only from \mathcal{G} . Let us define the *increment-distance* $\eta_{\ell, h}$ as

$$\eta_{\ell, h} \stackrel{\text{def}}{=} \min_{\substack{V \in \mathcal{C}(\ell, h) \\ V \neq e^{i\phi} \mathbb{I}}} d_{\mathbb{I}}(V). \quad (59)$$

Since \mathcal{G} is a finite gate set and $\mathcal{C}(\ell, h)$ has a bounded cardinality, $\eta_{\ell, h} > 0$ is a well-defined constant independent on n and represents the closest a circuit can be to being identity without being identity itself. This is formalized in the following fact.

► **Fact 15.** *Let V be a circuit $\in \mathcal{C}(\ell, h)$ such that $d_{\mathbb{I}}(V) < \eta_{\ell, h}$. There exists an angle ϕ_V such that $V = e^{i\phi_V} \mathbb{I}$.*

In order to construct a P algorithm for this problem, we notice that since $a = o(1)$, for some sufficiently large N_0 , if $n > N_0$

$$a(n) < \eta_{2^{t+1}, t}. \quad (60)$$

Our algorithm will solve only instance of this size or larger. Assume that an instance U of the problem is a False instance, so U is near-identity. Then for each qubit i and every state ψ ,

$$\left\| \text{tr}_{-i} \left(U_{L_i} (\psi_{L_i} \otimes \nu_{-L_i}) U_{L_i}^\dagger \right) - \psi_i \right\| \leq a \quad (61)$$

as a consequence of the prior stated facts. Since, this holds for all states ψ , then $d_{\mathbb{I}}(U_{L_i}) < a$. However, the circuit U_{L_A} acts on at most 2^{t+1} qubits and has depth at most t . Since $a < \eta_{2^{t+1}, t}$, then we can conclude that the action of U_{L_i} on the i th qubit must be \mathbb{I} (up to phase) in every False instance. Since this holds for every qubit i , in a False instance, the circuit U must exactly be \mathbb{I} (up to phase). Therefore, the P algorithm is simple: test if each circuit U_{L_i} is exactly identity and if so report False or otherwise report True. ◀

References

- 1 Scott Aaronson and Daniel Gottesman. Improved simulation of stabilizer circuits. *Physical Review A*, 70(5):052328, 2004.
- 2 Scott Aaronson and Greg Kuperberg. Quantum versus classical proofs and advice. In *Twenty-Second Annual IEEE Conference on Computational Complexity (CCC’07)*, pages 115–128, 2007. doi:10.1109/CCC.2007.27.
- 3 Dorit Aharonov and Tomer Naveh. Quantum NP-a survey. *arXiv quant-ph/0210077*, 2002.
- 4 F Barahona. On the computational complexity of Ising spin glass models. *Journal of Physics A: Mathematical and General*, 15(10):3241–3253, October 1982. doi:10.1088/0305-4470/15/10/028.
- 5 Sergey Bravyi and Alexei Kitaev. Universal quantum computation with ideal Clifford gates and noisy ancillas. *Physical Review A*, 71, March 2004. doi:10.1103/PhysRevA.71.022316.
- 6 Sergey Bravyi, Dan Browne, Padraic Calpin, Earl Campbell, David Gosset, and Mark Howard. Simulation of quantum circuits by low-rank stabilizer decompositions. *Quantum*, 3:181, September 2019. doi:10.22331/q-2019-09-02-181.
- 7 Sergey Bravyi, David Fattal, and Daniel Gottesman. GHZ extraction yield for multipartite stabilizer states. *Journal of Mathematical Physics*, 47(6):062106, 2006.
- 8 Sergey Bravyi and David Gosset. Improved classical simulation of quantum circuits dominated by Clifford gates. *Phys. Rev. Lett.*, 116:250501, June 2016. doi:10.1103/PhysRevLett.116.250501.
- 9 Sergey Bravyi, Graeme Smith, and John A. Smolin. Trading classical and quantum computational resources. *Phys. Rev. X*, 6:021043, June 2016. doi:10.1103/PhysRevX.6.021043.
- 10 David Fattal, Toby S Cubitt, Yoshihisa Yamamoto, Sergey Bravyi, and Isaac L Chuang. Entanglement in the stabilizer formalism. *arXiv preprint quant-ph/0406168*, 2004.
- 11 Bill Fefferman and Shelby Kimmel. Quantum vs. classical proofs and subset verification. In *43rd International Symposium on Mathematical Foundations of Computer Science, MFCS*, volume 117 of *LIPICs*, pages 22:1–22:23. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2018.
- 12 Hector J Garcia, Igor L Markov, and Andrew W Cross. Efficient inner-product algorithm for stabilizer states, 2012. arXiv:1210.6646.
- 13 Hector J Garcia-Ramirez. *Hybrid Techniques for Simulating Quantum Circuits using the Heisenberg Representation*. PhD thesis, University of Michigan, 2014.
- 14 Daniel Gottesman. The heisenberg representation of quantum computers. *Group22: Proceedings of the XXII International Colloquium on Group Theoretical Methods in Physics*, 1998. arXiv:arXiv:quant-ph/9807006.
- 15 Russell Impagliazzo and Ramamohan Paturi. On the Complexity of k-SAT. *Journal of Computer and System Sciences*, 62(2):367–375, March 2001. doi:10.1006/jcss.2000.1727.

- 16 Dominik Janzing, Pawel Wocjan, and Thomas Beth. “non-identity-check” is QMA-complete. *International Journal of Quantum Information*, 03(03):463–473, 2005. doi:10.1142/S0219749905001067.
- 17 Zhengfeng Ji and Xiaodi Wu. Non-identity check remains QMA-complete for short circuits. *arXiv preprint*, 2009. arXiv:0906.5416.
- 18 Alexei Yu Kitaev, Alexander Shen, Mikhail N Vyalyi, and Mikhail N Vyalyi. *Classical and quantum computation*. Number 47 in Graduate Studies in Mathematics. American Mathematical Soc., 2002.
- 19 Jacek Kuczyński and Henryk Woźniakowski. Estimating the largest eigenvalue by the power and Lanczos algorithms with a random start. *SIAM journal on matrix analysis and applications*, 13(4):1094–1122, 1992.
- 20 Ashley Montanaro and Ronald de Wolf. A survey of quantum property testing. *Theory Comput.*, 7:1–81, 2016. doi:10.4086/toc.gs.2016.007.
- 21 Tomoyuki Morimae, Masahito Hayashi, Harumichi Nishimura, and Keisuke Fujii. Quantum Merlin-Arthur with Clifford Arthur. *arXiv preprint*, 2015. arXiv:1506.06447.
- 22 Bryan O’Gorman. Parameterization of Tensor Network Contraction. In *14th Conference on the Theory of Quantum Computation, Communication and Cryptography (TQC 2019)*, volume 135 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 10:1–10:19, Dagstuhl, Germany, 2019. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.
- 23 Hammam Qassim, Hakop Pashayan, and David Gosset. Improved upper bounds on the stabilizer rank of magic states. *arXiv preprint*, 2021. arXiv:2106.07740.

A Alternative algorithm to Theorem 2

In Theorem 16, we give an alternative algorithm for solving the same problem as Theorem 2. While this algorithm will have a inferior worst-case runtime than Theorem 2, it may run significantly faster depending on the structure of the problem instance. Furthermore, it has the added advantage that its runtime can be efficiently calculated in time $\text{poly}(n, s)$. Therefore, one can quickly compute the runtime of Theorem 16 and compare it to that of Theorem 2 and run the faster algorithm. While not faster in a worst-case sense, it may prove optimal for many physical instances. In addition, Theorem 16 can handle not just T gates but all single-qubit phase gates $G = \text{diag}(1, e^{i\theta})$ which may be an advantage for some problems⁵.⁶

► **Theorem 16.** *For $t \leq n$, there exists a classical algorithm for parametrized QCSAT instance with a single-qubit output register consisting of Clifford and phase gates $G = \text{diag}(1, e^{i\theta})$, running in worst-case time $O(\text{poly}(n, s)2^{3t} \log(t/\delta))$, which decides if $\text{Val} > c$ or $< c - \delta$. Furthermore, there exists a classical $\text{poly}(n, s)$ routine to calculate the runtime of this algorithm without running the algorithm itself.*

Proof. Let us notice that our goal is to compute the largest eigenvalue of $\langle 0^m | U^\dagger |1\rangle\langle 1|_1 U |0^m\rangle$ due to Equation (1). Since $|1\rangle\langle 1| = \frac{\mathbb{I}}{2} - \frac{Z}{2}$, this is equivalent to computing the smallest eigenvalue of $\langle 0^m | U^\dagger Z_1 U |0^m\rangle$. In the case that U is a completely Clifford circuit, $Q^{(s)} = U^\dagger Z_1 U$ is a Pauli matrix $Q^{(s)} = \alpha P_1 \otimes P_2 \otimes \dots \otimes P_{n+m}$ for $\alpha \in \{\pm 1, \pm i\}$ and $P_i \in \{\mathbb{I}, X, Y, Z\}$.

⁵ It is also easy to extend this algorithm to all k -qubit non-Clifford gates. However, the runtime will now scale as 4^{t+kt} . This follows directly from the fact that any k -qubit non-Clifford gate can be expressed as the linear combination of 4^k Clifford gates.

⁶ Theorem 2 can also be extended to general phase gates, but at the cost of potentially weaker upper bounds on the constant α in the exponent.

Therefore,

$$\langle 0^m | U^\dagger Z U | 0^m \rangle = P_1 \otimes \dots \otimes P_n \cdot \prod_{j=n+1}^{n+m} \langle 0 | P_j | 0 \rangle. \quad (62)$$

Then, the smallest eigenvalue of this matrix is easy to calculate as it is in tensor product; this is effectively the Gottesman-Knill theorem [14]. Furthermore, the Pauli $P^{(s)}$ can be computed efficiently. More specifically, if $U = g_1 \dots g_s$ with each gate g_i a Clifford matrix, we can define and compute the sequence of Paulis $Q^{(k)} \stackrel{\text{def}}{=} g_k Q^{(k-1)} g_k^\dagger$ for $Q^{(0)} = Z_1$ from $k = 1, \dots, s$.

We now extend this algorithm to the case that U contains t non-Clifford gates. Consider first the case that there is one non-Clifford qubit rotation gate g_k , with g_k

$$g_k \stackrel{\text{def}}{=} R(\theta_k) = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\theta_k} \end{pmatrix} \quad (63)$$

and acts (without loss of generality) on the first qubit. Let $Q^{(k-1)}$ be the Pauli calculated up to gate g_{k-1} . Now notice that there are 2 cases to consider; namely if the action of $Q^{(k-1)}$ on the first qubit is $\in \{\mathbb{I}, Z\}$ or is $\in \{X, Y\}$. Since $R(\theta)$ commutes with \mathbb{I} and Z and the following commutation relations hold:

$$R(\theta) X R(\theta)^\dagger = (\cos \theta) X + (\sin \theta) Y, \quad R(\theta) Y R(\theta)^\dagger = (-\sin \theta) X + (\cos \theta) Y, \quad (64)$$

we can express

$$Q^{(k)} = g_k Q^{(k-1)} g_k^\dagger = P^{(k,1)} + P^{(k,2)} \quad (65)$$

where $P^{(k,1)}$ and $P^{(k,2)}$ are Pauli matrices scaled by a complex number. For every subsequent Clifford gate $g_{k'}$ we can then recursively define and compute

$$P^{(k',\ell)} \stackrel{\text{def}}{=} g_{k'} P^{(k'-1,\ell)} g_{k'}^\dagger \quad (66)$$

which will remain a Pauli matrix. It is easy to note that this bifurcation from one Pauli matrix to two Pauli matrices when commuting past a non-Clifford gate generalizes to multiple non-Clifford gates with

$$U Z_1 U^\dagger = Q^{(s)} = \sum_{\ell=1}^{\leq 2^t} P^{(s,\ell)} \quad (67)$$

being expressible as the linear combination of $\leq 2^t$ Pauli matrices each acting on $n + m$ qubits. We next show that although there are $\leq 2^t$ Pauli matrix, there exists an efficiently computable basis of size $b \leq t + 1$ such that each Pauli matrix can be expressed as a product of the basis matrices.

► **Lemma 17.** *Let U be a quantum circuit consisting of s gates of which at most t gates are non-Clifford qubit rotation gates. Then $Q^{(s)} = U Z_1 U^\dagger$ can be expressed as a sum of $\leq 2^b$ Pauli matrices which are each products of at most $b \leq t + 1$ basis Pauli matrices. Furthermore, the basis can be computed in time $O(\text{poly}(s))$ and the collection of Pauli matrices can be computed in time $O(2^b \cdot \text{poly}(s))$.*

3:18 The Parametrized Complexity of Quantum Verification

This lemma is proved after the description of the rest of the algorithm. Given the basis $\mathcal{B} = \{B^{(1)}, \dots, B^{(b)}\}$ for $b \leq t + 1$, define $\gamma(k', k) \stackrel{\text{def}}{=} 1$ if $B^{(k')}$ and $B^{(k)}$ commute and $\stackrel{\text{def}}{=} 0$ otherwise. Observe then the Pauli matrices

$$A^{(k)} \stackrel{\text{def}}{=} \prod_{k' < k} X_{k'}^{\gamma(k', k)} \cdot Z_k \quad (68)$$

observe the same commutation relations as \mathcal{B} do. However, $\mathcal{A} = \{A^{(1)}, \dots, A^{(b)}\}$ act on at most b qubits. For each Pauli matrix $P^{(\ell)}$ defined as a product of elements from \mathcal{B} , let us define $O^{(\ell)}$ as the same product, except using the corresponding matrices from \mathcal{A} . Then the spectrum of

$$H' \stackrel{\text{def}}{=} \sum_{\ell=1}^{\leq 2^t} O^{(\ell)} \quad (69)$$

is the same as that of $Q^{(s)}$ from Equation (67). This is because there exists a unitary mapping \mathcal{A} to \mathcal{B} which therefore maps $O^{(\ell)}$ to $P^{(\ell)}$ and likewise maps H to $Q^{(s)}$. Therefore, it suffices to compute the minimum eigenvalue of H' . Here H' is a square matrix of dimension $2^b \times 2^b$ with $b \leq t + 1$. Computing H' and its minimum eigenvalue to accuracy δ can be done in time $O(\text{poly}(s)2^{3b} \log(t/\delta))$.

Lastly, notice that a convenient quality of this algorithm is that the basis \mathcal{B} and its size b can be computed in time $O(\text{poly}(s))$. Therefore, one can calculate b and calculate⁷ the runtime of the algorithm without running the algorithm itself. ◀

Proof of Lemma 17. We proceed by induction on the gates g_1, \dots, g_s of U . Initially, the only basis matrix is $B^{(0,1)} = Z_1$ and $P^{(0,1)} = B^{(0,1)}$. Then in the inductive step, we assume a basis of $\{B^{(k,\lambda)}\}$ and a collection of Pauli matrices $P^{(k,\ell)}$ such that each Pauli is expressible as a product of the basis matrices. When gate g_k is a Clifford, we define $B^{(k-1,\lambda)} \stackrel{\text{def}}{=} g_k B^{(k-1,\lambda)} g_k^\dagger$. Conveniently, $P^{(k,\ell)} = g_k P^{(k-1,\ell)} g_k^\dagger$ is the product of the corresponding set of new basis matrices.

In the case that $g_k = R(\theta)$ which (without loss of generality) acts on the first qubit, we first transform the basis $\{B^{(k-1,\lambda)}\}$ by multiplying basis terms we ensure that at most 2 basis terms act non-trivially on the first qubit. The terms $P^{(k-1,\ell)}$ can be adjusted in polynomial-time to reflect the new basis. If there are no basis terms acting non-trivially or one basis term acting as Z , then we set $P^{(k,\ell)}$ equal to $P^{(k-1,\ell)}$. In the case that the one basis term (without loss of generality, $B^{(k-1,1)}$) acts as X , then we introduce a new basis term defined as $B^{(k,\text{new})} \stackrel{\text{def}}{=} B^{(k,1)} \cdot XY$. Any Pauli term $P^{(k-1,\ell)}$ involving $B^{(k-1,1)}$ after commuting by g_k now a linear combination of said term and $B^{(k,\lambda_{\text{new}})}$ according to Equation (64). A similar argument holds when the one basis $B^{(k-1,1)}$ term acts as Y . In the case that two basis terms act non-trivially on the first qubit, we can also enforce that one of the basis terms acts as Z and the other acts as either X or Y . Then, a similar argument enforces that it suffices to introduce a single additional basis term.

Therefore, at the end of the induction, the total basis has size at most $t + 1$ and UZ_1U^\dagger can be expressed as a linear combination of $\leq 2^t$ Pauli terms. ◀

⁷ Namely, this is to check in time $O(\text{poly}(s))$ if $b \ll t$ to see if this algorithm will be more efficient at computing H' than the stabilizer-rank of magic states algorithm (Theorem 2) derived from [23].

Averaged Circuit Eigenvalue Sampling

Steven T. Flammia  

AWS Center for Quantum Computing, Pasadena, CA, USA
California Institute of Technology, Pasadena, CA, USA

Abstract

We introduce ACES, a method for scalable noise metrology of quantum circuits that stands for Averaged Circuit Eigenvalue Sampling. It simultaneously estimates the individual error rates of all the gates in collections of quantum circuits, and can even account for space and time correlations between these gates. ACES strictly generalizes randomized benchmarking (RB), interleaved RB, simultaneous RB, and several other related techniques. However, ACES provides much more information and provably works under strictly weaker assumptions than these techniques. Finally, ACES is extremely scalable: we demonstrate with numerical simulations that it simultaneously and precisely estimates all the Pauli error rates on every gate and measurement in a 100 qubit quantum device using fewer than 20 relatively shallow Clifford circuits and an experimentally feasible number of samples. By learning the detailed gate errors for large quantum devices, ACES opens new possibilities for error mitigation, bespoke quantum error correcting codes and decoders, customized compilers, and more.

2012 ACM Subject Classification Hardware → Quantum computation

Keywords and phrases Quantum noise estimation, quantum benchmarking, QCVV

Digital Object Identifier 10.4230/LIPIcs.TQC.2022.4

Related Version *Averaged circuit eigenvalue sampling*: <https://arxiv.org/abs/2108.05803>

Supplementary Material The source code used to perform the simulations and generate the figures is available on GitHub:

Software (Source Code): <https://github.com/sflammia/ACES>

Funding This work was supported by the ARO QCISS program grant W911NF2110001.

Acknowledgements We thank Laura DeLorenzo, Robin Harper, Robert Huang, Alex Kubica, Ryan O'Donnell, Colm Ryan, Prasahnt Sivarajah, and Giacomo Torlai for discussions.

1 Introduction

Estimating errors in quantum computers is essential for progress towards fault tolerant quantum computation (FTQC) [36]. An error is any undesired quantum evolution, and so errors can be as general as the set of allowed quantum dynamics, making them difficult to estimate and characterize. The most relevant errors in the context of FTQC can be broadly cast into the two archetypes of coherent and incoherent errors [31], though this is not an exclusive dichotomy.

Coherent errors are roughly those that we wish to reduce through improved calibration or eliminate via dynamical decoupling [42], though clever choices of quantum codes and circuits can also be tailored to handle coherent noise [9, 22, 46]. These methods reach natural limits when the coherent noise becomes too complex to efficiently describe. While in principle coherent errors can accumulate badly during a computation [31], quantum error correction itself seems to reduce the coherence of noise [23, 3, 26].

Incoherent noise, by contrast, can generally only be completely fixed by quantum error correction and fault tolerance, though near-term strategies for error mitigation could also help [38, 32, 11, 8, 34]. Optimizing the codes, decoders, and circuits for FTQC requires a



© Steven T. Flammia;

licensed under Creative Commons License CC-BY 4.0

17th Conference on the Theory of Quantum Computation, Communication and Cryptography (TQC 2022).

Editors: François Le Gall and Tomoyuki Morimae; Article No. 4; pp. 4:1–4:10

Leibniz International Proceedings in Informatics



LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

comprehensive understanding of the incoherent noise in a quantum device. Many techniques have been developed to estimate incoherent errors, including randomized benchmarking (RB) [10], interleaved RB [35], simultaneous RB [17], character RB [21], and Pauli noise estimation [14] among others. Each of these techniques has in common that a general quantum noise source (which may include coherent errors) is actively *averaged* to obtain an incoherent noise model with the same fidelity using randomized control techniques [41, 28, 29, 43, 44]. It is this averaged noise that RB-type methods seek to estimate.

In this paper, we show that incoherent noise, modeled as a Pauli channel, can be learned extremely efficiently using averaged circuit eigenvalue sampling, or ACES. It is already known that Pauli channels can be (individually) estimated efficiently and in a manner that is robust to state preparation and measurement (SPAM) errors [19, 20, 14, 16], and they are effective at modeling noise for FTQC [18, 27]. ACES goes beyond this prior work and *simultaneously* estimates a large collection of Pauli noise channels associated to a quantum device. Indeed, we give numerical simulations showing that ACES can characterize every error rate associated to the Clifford gates in a 100 qubit quantum device using fewer than 20 circuits and a reasonable number of samples. Moreover, it requires only very simple classical resources to process these data, unlike other characterization techniques based on simulating or implementing general quantum circuits, or using challenging tensor network simulations [4, 5, 37, 39, 33, 6].

2 The Pauli and Clifford groups

The n -qubit *Pauli group* P_n consists of n -fold tensor products of single-qubit Pauli operators labeled as follows. Let \mathbf{a} be a $2n$ -bit string $\mathbf{a} = a_1 a_2 \dots a_{2n}$ and write $P_{\mathbf{a}} = i^{\mathbf{a}^T \Upsilon \mathbf{a}} \prod_{j=1}^n X_j^{a_{2j-1}} Z_j^{a_{2j}}$, where X_j and Z_j are single-qubit Paulis acting on qubit j , and $\Upsilon = \bigoplus_{k=1}^n \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$ is such that $P_{\mathbf{a}}$ is always hermitian. The group P_n contains these $P_{\mathbf{a}}$, together with the overall phases $\{\pm 1, \pm i\}$, composed under matrix multiplication. All elements of the Pauli group satisfy

$$P_{\mathbf{a}} P_{\mathbf{b}} = (-1)^{\langle \mathbf{a}, \mathbf{b} \rangle} P_{\mathbf{b}} P_{\mathbf{a}}, \quad (1)$$

where the sign is determined by the binary symplectic form $\langle \mathbf{a}, \mathbf{b} \rangle = \mathbf{a}^T (\Upsilon + \Upsilon^T) \mathbf{b} \bmod 2$.

The normalizer of the Pauli group inside the unitary group, modulo phases, is the *Clifford group* C_n , and it is generated by the controlled-NOT gate $CX_{i \rightarrow j}$ from control i to target j , the Hadamard gate H_j , and the phase gate S_j ¹.

Pauli channels are quantum channels of the form

$$\rho \mapsto \sum_{\mathbf{a}} p_{\mathbf{a}} P_{\mathbf{a}} \rho P_{\mathbf{a}}^{\dagger}, \quad (2)$$

where $p_{\mathbf{a}}$ is a (possibly subnormalized) probability distribution called the *Pauli error rates*. Leakage from the qubit space occurs when $\sum_{\mathbf{a}} p_{\mathbf{a}} < 1$. When a general quantum channel $\mathcal{E} = \sum_j K_j \cdot K_j^{\dagger}$ is *twirled* by the Pauli group, it becomes a Pauli channel denoted \mathcal{E}^P ,

$$\mathcal{E}^P(\rho) = \frac{1}{4^n} \sum_{\mathbf{a}} P_{\mathbf{a}}^{\dagger} \mathcal{E}(P_{\mathbf{a}} \rho P_{\mathbf{a}}^{\dagger}) P_{\mathbf{a}}. \quad (3)$$

¹ From this definition, elements of the Clifford group are actually equivalence classes up to an overall phase, but by a slight abuse of language we can speak about a “Clifford unitary” to mean any representative element up to a phase and refer to uniqueness of a Clifford unitary when we really mean uniqueness up to an overall phase.

If $K_j = \sum_{\mathbf{a}} \nu_{j,\mathbf{a}} P_{\mathbf{a}}$, then the Pauli error rates of \mathcal{E}^P are $p_{\mathbf{a}} = \sum_j |\nu_{j,\mathbf{a}}|^2$. Thus we can speak of the Pauli error rates of a general channel by considering its Pauli twirl. Note that we can interpret twirling as the mean of a random process where a Pauli is selected uniformly at random and applied both before and after the channel.

The eigenvectors of a Pauli channel \mathcal{E} are just the Pauli operators. Indeed, from Equation (1) we have $\mathcal{E}(P_{\mathbf{b}}) = \lambda_{\mathbf{b}} P_{\mathbf{b}}$ where the *Pauli eigenvalues* $\lambda_{\mathbf{b}}$ are,

$$\lambda_{\mathbf{b}} = \sum_{\mathbf{a}} (-1)^{\langle \mathbf{a}, \mathbf{b} \rangle} p_{\mathbf{a}}. \quad (4)$$

This equation can be inverted to express the error rates in terms of the eigenvalues ²,

$$p_{\mathbf{a}} = \frac{1}{2^n} \sum_{\mathbf{b}} (-1)^{\langle \mathbf{a}, \mathbf{b} \rangle} \lambda_{\mathbf{b}}. \quad (5)$$

We now introduce a “ \mathcal{G} -twisted” Pauli twirl. For a given Clifford \mathcal{G} and Pauli $P_{\mathbf{a}}$, let $P_{\mathbf{a}'} = \mathcal{G}(P_{\mathbf{a}})$. Note that since \mathcal{G} is unitary, the set of all \mathbf{a} and \mathbf{a}' are in one-to-one correspondence. We wish to expand a noisy gate as $\tilde{\mathcal{G}} = \mathcal{G}\mathcal{E}$ for some general noise channel $\mathcal{E} = \mathcal{G}^\dagger \tilde{\mathcal{G}}$. Intuitively, \mathcal{E} is close to the identity, though the definition doesn’t assume that. Then the \mathcal{G} -twisted twirl of $\tilde{\mathcal{G}}$ is

$$\tilde{\mathcal{G}}^{\mathcal{G}P}(\rho) = \frac{1}{4^n} \sum_{\mathbf{a}} P_{\mathbf{a}'}^\dagger \tilde{\mathcal{G}}(P_{\mathbf{a}} \rho P_{\mathbf{a}}^\dagger) P_{\mathbf{a}'} = \mathcal{G}(\mathcal{E}^P(\rho)). \quad (6)$$

From the last equality, we see that the \mathcal{G} -twisted twirl isolates the Pauli noise around a given noisy implementation $\tilde{\mathcal{G}}$ of an ideal Clifford gate \mathcal{G} .

\mathcal{G} -twisted twirled channels have an analogous eigendecomposition to a Pauli twirled channel, but with the notion of *generalized eigenvector*. Given such a channel $\tilde{\mathcal{G}}^{\mathcal{G}P}$, the generalized eigenvectors with respect to \mathcal{G}_0 are vectors such that $\tilde{\mathcal{G}}^{\mathcal{G}P}(v) = \lambda \mathcal{G}_0(v)$. We see from Equation (6) that choosing $\mathcal{G}_0 = \mathcal{G}$ gives exactly the Paulis as the generalized eigenvectors with eigenvalues given by the Pauli eigenvalues of the noise map \mathcal{E}^P .

3 Averaged circuits

Let us consider a Clifford circuit (i.e., a circuit composed solely of CX , H , and S gates or an equivalent generating set), denoted \mathcal{C} . Any physical implementation of these circuits will be noisy, and we seek to characterize the incoherent Pauli-averaged noise in these circuits, specifically in the generators used to create the circuits. To that end, from the circuit \mathcal{C} we create a new ensemble of circuits \mathcal{C}^P by sampling a \mathcal{G} -twisted Pauli twirl across each Clifford circuit element and recompiling the Pauli gate. This approach to Pauli frame randomization is known as randomized compiling [43]. Each circuit in the ensemble implements the same unitary operation, but now the noise has been averaged over the Pauli group. In Ref. [43], it was proven that circuits sampled in this way yield on average a circuit that interleaves Pauli-averaged noise with ideal gates (except possibly in the final measurement step). This result rigorously holds whenever the noise on each Pauli gate is the same, and furthermore Ref. [43] provides some robustness guarantees in the event that this assumption is perturbatively violated ³.

² Note that $\lambda_{\mathbf{a}}$ and $p_{\mathbf{a}}$ are essentially Fourier transform pairs since the transform relating them is the Walsh-Hadamard transform (up to a permutation). A helpful intuition is that λ lives in the “time domain” where we can efficiently sample, and p lives in the “frequency domain” where we wish to reconstruct the signal.

³ While the gate-independent noise assumption may seem unrealistic, it should be noted that the successful and widely used method of standard RB makes the *much* stronger assumption that the noise is gate-independent across all n -qubit Clifford gates, whereas ACES weakens this substantially to just the Pauli gates.

4:4 Averaged Circuit Eigenvalue Sampling

These considerations motivate considering only *averaged circuits*, denoted $\mathcal{C}^{\mathbb{P}}$, so that the noisy physical implementations will have the form

$$\widetilde{\mathcal{C}}^{\mathbb{P}} = \widetilde{\mathcal{G}}_T^{\mathcal{G}_T^{\mathbb{P}}} \dots \widetilde{\mathcal{G}}_1^{\mathcal{G}_1^{\mathbb{P}}} = \mathcal{G}_T \mathcal{E}_{\mathcal{G}_T} \dots \mathcal{G}_1 \mathcal{E}_{\mathcal{G}_1}. \quad (7)$$

4 Eigenvalue sampling

Let us suppose for the moment that a given circuit \mathcal{C} ideally implements the identity unitary. Under the gate-independent noise assumption, it follows that the noisy implementation of the averaged circuit, $\widetilde{\mathcal{C}}^{\mathbb{P}}$, will be a Pauli channel. It therefore has Pauli eigenvalues, namely $\widetilde{\mathcal{C}}^{\mathbb{P}}(P_{\mathbf{a}}) = \Lambda_{\mathcal{C},\mathbf{a}} P_{\mathbf{a}}$, where we use capital Λ to denote this circuit-level eigenvalue. Because of the gate-independent noise assumption, this eigenvalue depends only on the eigenvector ($P_{\mathbf{a}}$) and on the circuit (\mathcal{C}), so it is labeled accordingly as $\Lambda_{\mathcal{C},\mathbf{a}}$.

If the circuit \mathcal{C} does not implement the identity unitary, but rather some net Clifford operation, something similar still holds. If the ideal circuit maps an input Pauli $P_{\mathbf{a}}$ to an output Pauli $\mathcal{C}(P_{\mathbf{a}}) = \pm P_{\mathbf{a}'}$, then the overall \pm sign and the value of \mathbf{a}' can be efficiently computed [1]. The noisy version of the circuit will give an averaged operator that satisfies the generalized eigenvalue equation

$$\widetilde{\mathcal{C}}^{\mathbb{P}}(P_{\mathbf{a}}) = \Lambda_{\mathcal{C},\mathbf{a}} \mathcal{C}(P_{\mathbf{a}}) = \pm \Lambda_{\mathcal{C},\mathbf{a}} P_{\mathbf{a}'}. \quad (8)$$

From the orthogonality of the Pauli basis, it follows that

$$\Lambda_{\mathcal{C},\mathbf{a}} = \pm \frac{1}{2^n} \text{Tr}(P_{\mathbf{a}'} \widetilde{\mathcal{C}}^{\mathbb{P}}(P_{\mathbf{a}})), \quad (9)$$

and this suggests a prescription for estimating the (generalized) eigenvalue $\Lambda_{\mathcal{C},\mathbf{a}}$ that we call eigenvalue sampling.

To estimate $\Lambda_{\mathcal{C},\mathbf{a}}$ via eigenvalue sampling, let us focus on the case where $P_{\mathbf{a}}$ is a single-qubit Pauli. We begin by selecting uniformly at random an eigenstate ψ_{\pm} on the support of $P_{\mathbf{a}}$ having eigenvalue ± 1 (ignoring the other registers). Then we send ψ_{\pm} into a randomly chosen element in the circuit ensemble $\mathcal{C}^{\mathbb{P}}$ and measure the output in the basis defined by $P_{\mathbf{a}'}$. Our overall estimate for $\Lambda_{\mathcal{C},\mathbf{a}}$ consists of measuring N independent experiments and taking the difference of the sample averages between the ψ_+ and ψ_- experiments. It is easy to check that this differencing trick makes Equation (9) hold in expectation, so this is an unbiased estimator of $\Lambda_{\mathcal{C},\mathbf{a}}$. This sampling strategy was first analyzed in Ref. [15], and it is straightforward to generalize to the n -qubit case. Note that it will be most efficient if the support of $P_{\mathbf{a}}$ and $P_{\mathbf{a}'}$ are relatively small, and also that Paulis with disjoint support can implement such measurements simultaneously.

5 Relating circuit and gate eigenvalues

We have seen how eigenvalue sampling on averaged circuits gives us access to the (generalized) Pauli eigenvalues $\Lambda_{\mathcal{C},\mathbf{a}}$ in the implemented circuit ensemble $\widetilde{\mathcal{C}}^{\mathbb{P}}$. This is already a useful method for estimating the quality of the circuit implementation $\widetilde{\mathcal{C}}$, since it can be interpreted as a fidelity-like measure for how faithfully the circuit executes given the input $P_{\mathbf{a}}$. However, we seek to characterize not just the global circuit noise, but the error rates associated to the constituent gates as well. How do the (generalized) eigenvalues of the individual gates relate to the eigenvalue of the total circuit $\mathcal{C} = \mathcal{G}_T \dots \mathcal{G}_1$?

Let us apply the generalized eigenvalue relation sequentially to the gates in a Clifford circuit. For the first gate we obtain $\tilde{\mathcal{G}}_1^{\mathcal{P}}(P_{\mathbf{a}_1}) = \lambda_{1,\mathbf{a}_1} \mathcal{G}_1(P_{\mathbf{a}_1}) = (\pm)_1 \lambda_{1,\mathbf{a}_1} P_{\mathbf{a}_2}$. Acting on this with $\tilde{\mathcal{G}}_2^{\mathcal{P}}$, we obtain

$$\begin{aligned} \tilde{\mathcal{G}}_2^{\mathcal{P}} \tilde{\mathcal{G}}_1^{\mathcal{P}}(P_{\mathbf{a}_1}) &= (\pm)_1 (\pm)_2 \lambda_{1,\mathbf{a}_1} \lambda_{2,\mathbf{a}_2} \mathcal{G}_2^{\mathcal{P}}(P_{\mathbf{a}_2}) \\ &= (\pm)_1 (\pm)_2 \lambda_{1,\mathbf{a}_1} \lambda_{2,\mathbf{a}_2} P_{\mathbf{a}_3}. \end{aligned}$$

Continuing in this fashion, we find that

$$\tilde{\mathcal{C}}^{\mathcal{P}}(P_{\mathbf{a}_1}) = (\pm) \prod_{k=1}^T \lambda_{k,\mathbf{a}_k} \mathcal{C}(P_{\mathbf{a}_1}), \quad (10)$$

where the overall sign and the individual \mathbf{a}_k can be computed efficiently [1]. Comparing with Equation (8), we see that $\Lambda_{\mathcal{C},\mathbf{a}_1} = (\pm) \prod_k \lambda_{k,\mathbf{a}_k}$. We will use the freedom to reinterpret the sign of the input Pauli $P_{\mathbf{a}_1}$ to ensure that we always have a + sign in this equation, and therefore we have the relation

$$\Lambda_{\mathcal{C},\mathbf{a}_1} = \prod_{k=1}^T \lambda_{k,\mathbf{a}_k}. \quad (11)$$

With this sign convention, in the regime of interest $\Lambda_{\mathcal{C},\mathbf{a}_1}$ is positive and not too small. We therefore focus on sets of circuits \mathcal{C}_k and input labels \mathbf{a}_{k_i} such that $\Lambda_{\mathcal{C}_k,\mathbf{a}_{k_i}}$ is always larger than, say, $1/2$, and gates where $\lambda_{k,\mathbf{a}_k} > 0$.

6 Estimating gate errors via ACES

We now consider a circuit \mathcal{C}_k and an input label \mathbf{a}_{k_i} ; we give this combination a composite index $\mu = (\mathcal{C}_k, \mathbf{a}_{k_i})$. From the above discussion, we can obtain an empirical estimate $\hat{\Lambda}_\mu$ of Λ_μ by eigenvalue sampling on the averaged circuit ensemble for the circuit/input label μ . Similarly, we assemble all gate-level eigenvalues under a single index to get λ_ν , where ν labels pairs of gates and Paulis whose noise we wish to model. Since all eigenvalue quantities are positive in the regime of interest, we can introduce new variables,

$$\Lambda_\mu = e^{-b_\mu}, \quad \lambda_\nu = e^{-x_\nu}. \quad (12)$$

The new variables are related by the *linear* equations

$$\sum_{\nu} A_{\mu\nu} x_\nu = b_\mu. \quad (13)$$

We refer to the matrix A as the *design matrix*. Once enough independent equations (μ) are obtained so that A is full rank, an estimate for \mathbf{x} can be obtained in any number of ways, most straightforwardly via least squares as $\hat{\mathbf{x}} = A^+ \hat{\mathbf{b}}$, where $\hat{\mathbf{b}}$ denotes an empirical estimate for \mathbf{b} and A^+ is the pseudoinverse of A . Inverting Equation (12) subsequently gives us estimates for λ_ν , and Pauli error rates can be obtained by using Equation (5).

The precision of our estimate depends in part on the choice of A , as well as the precision of the initial estimates of the Λ_μ . The estimates for λ_ν are always accurate in the sense that these are consistent estimators, however they will in general have some bias. In the numerical simulations below, no attempt was made to find optimal designs A , and only random choices were used. We leave open the question of finding optimal design matrices that maximize the precision and accuracy of these estimators.

4:6 Averaged Circuit Eigenvalue Sampling

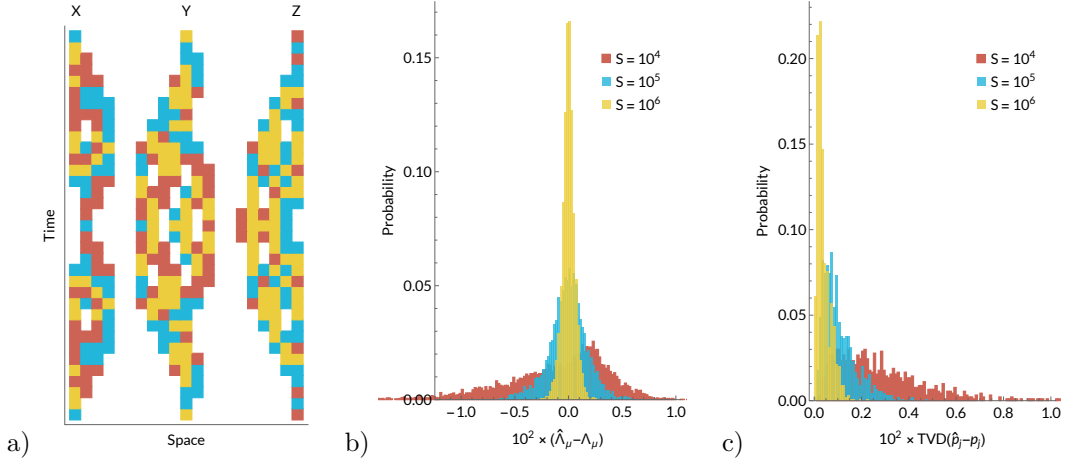


Figure 1 a) Sending X , Y , and Z Paulis (blue, yellow, and red, resp.) through a small “mirror circuit” (i.e., one of the form UU^\dagger) with $n = 21$ qubits, depth $d = 34$, and nearest-neighbor gates in 1D. Normalized histograms of b) the absolute error for the μ th estimated circuit eigenvalue $\hat{\Lambda}_\mu$, and c) the total variation distance (TVD) for the estimated Pauli error rates \hat{p}_j of the noisy gate \hat{G}_j in a $n = 100$ qubit simulation. There are 898 gates (including measurements) in the model, and 10,155 estimated circuit fidelities (which are estimated in large batches due to the n -bit measurements) to estimate $N = 5070$ parameters. Plots are for a number of samples S per Λ_μ of 10^4 , 10^5 , and 10^6 .

7 Correlations and SPAM

The ACES methodology is flexible enough to allow independent estimation of SPAM errors as well as space- and/or certain time-correlated errors. To estimate measurement noise, we simply add a list of variables x_ν associated to each Pauli measurement error that we wish to model. We caution that separating preparation errors from measurement errors will not be possible if they are introduced into the model in a symmetric way (because then A will not be full rank); this problem is not unique to ACES however [4] and we do not attempt to resolve it here.

To handle space-correlated errors, we reinterpret the gates that generate our circuits to come in correlated groups. For example, if we want to model correlated noise between the Hadamard gates H_1 and H_2 , we could have separate variables for the gates H_1 , H_2 , and $H_1 \otimes H_2$. This is analogous to interleaved [35] and simultaneous RB [17], except that all of the data are used to fit all of the gates and correlations symmetrically and simultaneously.

Limited forms of time-correlated errors can be handled similarly by introducing variables for pairs of gates in time. For example, if the noise on H_1 depends on whether S_1 was applied or not right before, then we can introduce separate variables for these cases.

The only condition for a unique and consistent estimate in all of these scenarios is that the design matrix A is full rank. If A were random, then we only need as many equations as unknowns for this to hold with high probability. From this heuristic, we expect that the number of experiments should be about as large as, or a little larger than, the number of variables.

8 Numerical results

We now demonstrate the scalability of ACES via numerical simulations. Rigorous proofs of the consistency of ACES and bounds on the sample complexity will be presented elsewhere.

We consider the most general model of inhomogeneous but uncorrelated noise, plus readout errors⁴. In this model on n qubits there are $O(n)$ variables: CX gates acting between neighbors, together with six single-qubit Clifford gates (modulo the Paulis), and independent readout errors on each qubit in each Pauli basis.

We generated $C = 19$ random 1D Clifford circuits on $n = 100$ qubits of varying depths from $d = 2$ up to $d = 89$. The sum of all the circuit depths, including the measurement rounds, was 354. We then computed the circuit eigenvalues obtained from sending in all single-qubit Paulis and, on some circuits, two-qubit Paulis on nearest neighbors as well. We found it challenging to generate a full-rank design matrix A using the “mirror circuits” shown in Figure 1a, so we padded each mirror circuit with a depth 5 random circuit layer at the end. This means that the Paulis measured at the output had, in some cases, weight as high as 6, though most still had weight 1 or 2. Constant-weight Pauli operators can nonetheless be estimated efficiently from single-qubit Pauli measurements [12, 24, 25], and this only increases the sample complexity by a constant factor. We then generated a “true” noise model by assigning to each gate random Pauli error rates consistent with the estimates reported in the Arute et al. experiment [2]. The entire implementation can be found in the associated Mathematica notebook accompanying this manuscript [13].

Despite its seeming simplicity, this model still has $N = 5070$ parameters. Even under the simplifying assumptions of RB with Clifford averaging where the noise is depolarizing on each gate, there would still be 798 parameters (neglecting SPAM) to be estimated through interleaved RB, and even then the required Clifford randomizations would be prohibitively expensive.

ACES estimates all of these parameters with just these 19 random circuits (and their Pauli randomizations). This is possible because each measurement is an n -bit measurement, so many parameters are estimated in parallel.

In Figs. 1b and 1c we plot the convergence of the ACES estimate as a function of S , the number of samples per circuit eigenvalue estimate. Estimates \hat{x}_ν of the model parameters x_ν were obtained from the simulated data by solving Equation (13) with the simplest possible estimator, a truncated least-squares estimate (i.e., finding the least squares solution and truncating any negative values).

Counting an n -bit measurement as one sample, the total sample complexity is $O(SC)$ where C is the number of different averaged circuits used, in this case $C = 19$. Results are shown for $S = 10^4, 10^5, 10^6$. Even for $S = 10^4$, nearly all circuit eigenvalue estimates (1b) are within 1% of the true answer, and the total variation distance (TVD) between the estimated and true Pauli error rates on each gate are within .64% with 95% confidence. This latter figure improves to .1% with 95% confidence for $S = 10^6$, a remarkably precise estimate given how many parameters there are and that no regularization was used to avoid potential overfitting.

9 Discussion

There are many potential applications for ACES, and many avenues for improving and extending it as well. For example, in addition to the tailored codes and decoders mentioned above, error mitigation is one of the most natural applications of ACES [38, 32, 11, 8, 34], and it can also be used to debias estimates of classical shadows following the ideas in

⁴ We do not model state preparation errors, only readout errors, for the reason discussed in the main text that these are not separately identifiable.

Refs. [7, 30, 40]. Regarding extensions, while we have focused entirely on Clifford gates, it is easy to see that ACES can accommodate circuits with a constant number of T gates in specific configurations. However, extending beyond this to universal gate sets in general is an important question for future research. A differential analysis suggests that obtaining circuit eigenvalue estimates such that $\hat{\Lambda}_\mu = \Lambda_\mu \pm \epsilon \Lambda_\mu$ suffices to obtain gate-level eigenvalue estimates of order $\hat{\lambda}_\nu = \lambda_\nu \pm O(\|A^+\| \epsilon) \lambda_\nu$. Thus, finding circuits, Pauli inputs, and noise models whose associated design matrix minimizes $\|A^+\|$ could help optimize the sample efficiency of ACES. There are additional desiderata for the design matrix, such as requiring only few experiments and using circuits that map few-qubit Paulis to few-qubit Paulis. Finding a general understanding of which circuits behave best is an open question. While ACES can test for correlations in a given noise model, it would be more powerful to include a large model and then search for dominant correlations by enforcing sparsity. One way forward might be to test clusters of gates for inclusion using methods such as group LASSO [45]. Finally, the most obvious open problem is to implement ACES in a near-term experiment.




References

- 1 S. Aaronson and D. Gottesman. Improved simulation of stabilizer circuits. *Phys. Rev. A*, 70(5):052328, November 2004. doi:10.1103/PhysRevA.70.052328.
- 2 Frank Arute, Kunal Arya, Ryan Babbush, Dave Bacon, Joseph C. Bardin, Rami Barends, Rupak Biswas, Sergio Boixo, Fernando G. S. L. Brandao, David A. Buell, Brian Burkett, Yu Chen, Zijun Chen, Ben Chiaro, Roberto Collins, William Courtney, Andrew Dunsworth, Edward Farhi, Brooks Foxen, Austin Fowler, Craig Gidney, Marissa Giustina, Rob Graff, Keith Guerin, Steve Habegger, Matthew P. Harrigan, Michael J. Hartmann, Alan Ho, Markus Hoffmann, Trent Huang, Travis S. Humble, Sergei V. Isakov, Evan Jeffrey, Zhang Jiang, Dvir Kafri, Kostyantyn Kechedzhi, Julian Kelly, Paul V. Klimov, Sergey Knysh, Alexander Korotkov, Fedor Kostritsa, David Landhuis, Mike Lindmark, Erik Lucero, Dmitry Lyakh, Salvatore Mandrà, Jarrod R. McClean, Matthew McEwen, Anthony Megrant, Xiao Mi, Kristel Michielsen, Masoud Mohseni, Josh Mutus, Ofer Naaman, Matthew Neeley, Charles Neill, Murphy Yuezhen Niu, Eric Ostby, Andre Petukhov, John C. Platt, Chris Quintana, Eleanor G. Rieffel, Pedram Roushan, Nicholas C. Rubin, Daniel Sank, Kevin J. Satzinger, Vadim Smelyanskiy, Kevin J. Sung, Matthew D. Trevithick, Amit Vainsencher, Benjamin Villalonga, Theodore White, Z. Jamie Yao, Ping Yeh, Adam Zalcman, Hartmut Neven, and John M. Martinis. Quantum supremacy using a programmable superconducting processor. *Nature*, 574(7779):505–510, October 2019. doi:10.1038/s41586-019-1666-5.
- 3 Stefanie J. Beale, Joel J. Wallman, Mauricio Gutiérrez, Kenneth R. Brown, and Raymond Laflamme. Coherence in quantum error-correcting codes. *Phys. Rev. Lett.*, 121:190501, 2018. doi:10.1103/PhysRevLett.121.190501.
- 4 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:, 2016. doi:10.1038/ncomms14485.
- 5 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, April 2018. doi:10.1038/s41567-018-0124-x.
- 6 Haoyuan Cai, Qi Ye, and Dong-Ling Deng. Sample complexity of learning quantum circuits, 2021. arXiv:2107.09078.
- 7 Senrui Chen, Wenjun Yu, Pei Zeng, and Steven T. Flammia. Robust shadow estimation, 2020. arXiv:2011.09636.

- 8 Piotr Czarnik, Andrew Arrasmith, Patrick J. Coles, and Lukasz Cincio. Error mitigation with Clifford quantum-circuit data, 2021. [arXiv:2005.10189](https://arxiv.org/abs/2005.10189).
- 9 Dripto M. Debroy, Muyuan Li, Michael Newman, and Kenneth R. Brown. Stabilizer slicing: Coherent error cancellations in low-density parity-check stabilizer codes. *Phys. Rev. Lett.*, 121(25):250502, December 2018. doi:10.1103/physrevlett.121.250502.
- 10 Joseph Emerson, Robert Alicki, and Karol Życzkowski. Scalable noise estimation with random unitary operators. *J. Opt. B*, 7(10):S347, 2005. doi:10.1088/1464-4266/7/10/021.
- 11 Suguru Endo, Simon C. Benjamin, and Ying Li. Practical quantum error mitigation for near-future applications. *Phys. Rev. X*, 8(3):031027, July 2018. doi:10.1103/physrevx.8.031027.
- 12 Tim J. Evans, Robin Harper, and Steven T. Flammia. Scalable Bayesian Hamiltonian learning, 2019. [arXiv:1912.07636](https://arxiv.org/abs/1912.07636).
- 13 S. T. Flammia. ACES. <https://github.com/sflammia/ACES>, 2021.
- 14 Steven Flammia and Joel Wallman. Efficient estimation of Pauli channels. *ACM Transactions on Quantum Computing*, 1(1):1–32, 2020. doi:10.1145/3408039.
- 15 Steven T. Flammia and Yi-Kai Liu. Direct fidelity estimation from few Pauli measurements. *Phys. Rev. Lett.*, 106(23):230501, June 2011. doi:10.1103/PhysRevLett.106.230501.
- 16 Steven T. Flammia and Ryan O’Donnell. Pauli error estimation via population recovery. *Quantum*, 5:549, September 2021. doi:10.22331/q-2021-09-23-549.
- 17 Jay M. Gambetta, A. D. Córcoles, S. T. Merkel, B. R. Johnson, John A. Smolin, Jerry M. Chow, Colm A. Ryan, Chad Rigetti, S. Poletto, Thomas A. Ohki, Mark B. Ketchen, and M. Steffen. Characterization of addressability by simultaneous randomized benchmarking. *Phys. Rev. Lett.*, 109:240504, December 2012. doi:10.1103/PhysRevLett.109.240504.
- 18 Michael R. Geller and Zhongyuan Zhou. Efficient error models for fault-tolerant architectures and the Pauli twirling approximation. *Phys. Rev. A*, 88(1):012314, July 2013. doi:10.1103/physreva.88.012314.
- 19 Robin Harper, Steven T. Flammia, and Joel J. Wallman. Efficient learning of quantum noise. *Nature Physics*, 16(12):1184–1188, August 2020. doi:10.1038/s41567-020-0992-8.
- 20 Robin Harper, Wenjun Yu, and Steven T. Flammia. Fast estimation of sparse quantum noise. *PRX Quantum*, 2(1):010322, February 2021. doi:10.1103/prxquantum.2.010322.
- 21 Jonas Helsen, Xiao Xue, Lieven MK Vandersypen, and Stephanie Wehner. A new class of efficient randomized benchmarking protocols. *npj Quantum Information*, 5(1):71, August 2019. doi:10.1038/s41534-019-0182-7.
- 22 Jingzhen Hu, Qingzhong Liang, Narayanan Rengaswamy, and Robert Calderbank. Mitigating coherent noise by balancing weight-2 Z -stabilizers, 2021. [arXiv:2011.00197](https://arxiv.org/abs/2011.00197).
- 23 Eric Huang, Andrew C. Doherty, and Steven Flammia. Performance of quantum error correction with coherent errors. *Phys. Rev. A*, 99:022313, 2019. doi:10.1103/PhysRevA.99.022313.
- 24 Hsin-Yuan Huang, Richard Kueng, and John Preskill. Predicting many properties of a quantum system from very few measurements. *Nature Physics*, 16(10):1050–1057, June 2020. doi:10.1038/s41567-020-0932-7.
- 25 Hsin-Yuan Huang, Richard Kueng, and John Preskill. Efficient estimation of Pauli observables by derandomization, 2021. [arXiv:2103.07510](https://arxiv.org/abs/2103.07510).
- 26 Joseph K Iverson and John Preskill. Coherence in logical quantum channels. *New Journal of Physics*, 22(7):073066, August 2020. doi:10.1088/1367-2630/ab8e5c.
- 27 Amara Katabarwa and Michael R. Geller. Logical error rate in the Pauli twirling approximation. *Scientific Reports*, 5(1), September 2015. doi:10.1038/srep14670.
- 28 O. Kern, G. Alber, and D. L. Shepelyansky. Quantum error correction of coherent errors by randomization. *Euro. Phys. J. D*, 32(1):153–156, January 2005. doi:10.1140/epjd/e2004-00196-9.
- 29 E. Knill. Quantum computing with realistically noisy devices. *Nature*, 434(7029):39–44, March 2005. doi:10.1038/nature03350.
- 30 Dax Enshan Koh and Sabee Grewal. Classical shadows with noise, 2020. [arXiv:2011.11580](https://arxiv.org/abs/2011.11580).

- 31 Richard Kueng, David M. Long, Andrew C. Doherty, and Steven T. Flammia. Comparing experiments to the fault-tolerance threshold. *Phys. Rev. Lett.*, 117:170502, October 2016. doi:10.1103/PhysRevLett.117.170502.
- 32 Ying Li and Simon C. Benjamin. Efficient variational quantum simulator incorporating active error minimization. *Phys. Rev. X*, 7(2):021050, June 2017. doi:10.1103/physrevx.7.021050.
- 33 Yunchao Liu, Matthew Otten, Roozbeh Bassirianjahromi, Liang Jiang, and Bill Fefferman. Benchmarking near-term quantum computers via random circuit sampling, 2021. arXiv:2105.05232.
- 34 Angus Lowe, Max Hunter Gordon, Piotr Czarnik, Andrew Arrasmith, Patrick J. Coles, and Lukasz Cincio. Unified approach to data-driven quantum error mitigation, 2020. arXiv:2011.01157.
- 35 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. *Phys. Rev. Lett.*, 109:080505, August 2012. doi:10.1103/PhysRevLett.109.080505.
- 36 John M. Martinis. Qubit metrology for building a fault-tolerant quantum computer. *npj Quantum Information*, 1:, October 2015. doi:10.1038/npjqi.2015.5.
- 37 Runzhou Tao, Yunong Shi, Jianan Yao, John Hui, Frederic T. Chong, and Ronghui Gu. Gleipnir: Toward practical error analysis for quantum programs. In *Proceedings of the 42nd ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI 2021)*, 2021. doi:10.1145/3453483.3454029.
- 38 Kristan Temme, Sergey Bravyi, and Jay M. Gambetta. Error mitigation for short-depth quantum circuits. *Phys. Rev. Lett.*, 119(18):180509, November 2017. doi:10.1103/physrevlett.119.180509.
- 39 Giacomo Torlai, Christopher J. Wood, Atithi Acharya, Giuseppe Carleo, Juan Carrasquilla, and Leandro Aolita. Quantum process tomography with unsupervised learning and tensor networks, 2020. arXiv:2006.02424.
- 40 Ewout van den Berg, Zlatko K. Mineev, and Kristan Temme. Model-free readout-error mitigation for quantum expectation values, 2021. arXiv:2012.09738.
- 41 Lorenza Viola and Emanuel Knill. Random decoupling schemes for quantum dynamical control and error suppression. *Phys. Rev. Lett.*, 94:060502, February 2005. doi:10.1103/PhysRevLett.94.060502.
- 42 Lorenza Viola, Emanuel Knill, and Seth Lloyd. Dynamical decoupling of open quantum systems. *Phys. Rev. Lett.*, 82:2417–2421, March 1999. doi:10.1103/PhysRevLett.82.2417.
- 43 Joel J. Wallman and Joseph Emerson. Noise tailoring for scalable quantum computation via randomized compiling. *Phys. Rev. A*, 94:052325, November 2016. doi:10.1103/PhysRevA.94.052325.
- 44 Matthew Ware, Guilhem Ribeill, Diego Ristè, Colm A. Ryan, Blake Johnson, and Marcus P. da Silva. Experimental Pauli-frame randomization on a superconducting qubit. *Phys. Rev. A*, 103(4):042604, April 2021. doi:10.1103/physreva.103.042604.
- 45 Ming Yuan and Yi Lin. Model selection and estimation in regression with grouped variables. *J. R. Statist. Soc. B*, 68(1):49–67, 2006. doi:10.1111/j.1467-9868.2005.00532.x.
- 46 Bichen Zhang, Swarnadeep Majumder, Pak Hong Leung, Stephen Crain, Ye Wang, Chao Fang, Dripto M. Debroy, Jungsang Kim, and Kenneth R. Brown. Hidden inverses: Coherent error cancellation at the circuit level, 2021. arXiv:2104.01119.

Classical Simulation of Quantum Circuits with Partial and Graphical Stabiliser Decompositions

Aleks Kissinger   

University of Oxford, UK

John van de Wetering   

Radboud University Nijmegen, The Netherlands

University of Oxford, United Kingdom

Renaud Vilmart   

Université Paris-Saclay, CNRS, ENS Paris-Saclay, Inria,

Laboratoire Méthodes Formelles, 91190, Gif-sur-Yvette, France

Abstract

Recent developments in classical simulation of quantum circuits make use of clever decompositions of chunks of magic states into sums of efficiently simulable stabiliser states. We show here how, by considering certain non-stabiliser entangled states which have more favourable decompositions, we can speed up these simulations. This is made possible by using the ZX-calculus, which allows us to easily find instances of these entangled states in the simplified diagram representing the quantum circuit to be simulated. We additionally find a new technique of *partial* stabiliser decompositions that allow us to trade magic states for stabiliser terms. With this technique we require only $2^{\alpha t}$ stabiliser terms, where $\alpha \approx 0.396$, to simulate a circuit with T-count t . This matches the α found by Qassim et al. [16], but whereas they only get this scaling in the asymptotic limit, ours applies for a circuit of any size. Our method builds upon a recently proposed scheme for simulation combining stabiliser decompositions and optimisation strategies implemented in the software QuiZX [15]. With our techniques we manage to reliably simulate 50-qubit 1400 T-count hidden shift circuits in a couple of minutes on a consumer laptop.

2012 ACM Subject Classification Theory of computation \rightarrow Quantum computation theory

Keywords and phrases ZX-calculus, Stabiliser Rank, Quantum Simulation

Digital Object Identifier 10.4230/LIPIcs.TQC.2022.5

Related Version *Preprint*: <https://arxiv.org/abs/2202.09202>

Funding *Aleks Kissinger*: Acknowledges support from AFOSR grant FA2386-18-1-4028.

John van de Wetering: Supported by an NWO Rubicon Personal Grant.

Acknowledgements JvdW and AK would like to thank James Hefford for useful discussions regarding decompositions of cat states in the ZX-calculus.

1 Introduction

A landmark result in the study of quantum circuit simulation is the Gottesman-Knill theorem [1], which states that a quantum circuit consisting of Clifford gates and stabiliser state inputs can be efficiently classically simulated. While Clifford gates are not universal for quantum computation, the Clifford+ T gate set, where we also allow the single qubit T gate, can approximate any qubit unitary to arbitrary precision [2, 17]. It is widely believed that the Clifford+ T gate set requires exponentially large classical resources to simulate in general. However, in practice certain methods exist for simulating Clifford+ T circuits significantly larger than one could ever hope to simulate directly, e.g. by state-vector calculation.



© Aleks Kissinger, John van de Wetering, and Renaud Vilmart;
licensed under Creative Commons License CC-BY 4.0

17th Conference on the Theory of Quantum Computation, Communication and Cryptography (TQC 2022).

Editors: François Le Gall and Tomoyuki Morimae; Article No. 5; pp. 5:1–5:13

Leibniz International Proceedings in Informatics



LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

A particularly effective technique for circuits with relatively low numbers of non-Clifford gates is simulation by *stabiliser decomposition*. That is, general states (typically prepared via a Clifford+T circuit) are first decomposed into weighted sums of stabiliser terms, each of which can be simulated efficiently via Gottesman-Knill. The *stabiliser rank* of a state $|\psi\rangle$ is the number of terms in the smallest possible decomposition of $|\psi\rangle$ as a weighted sum of stabiliser states. While computing the exact stabiliser rank of a given state is expected to be a hard problem, various heuristics exist for obtaining stabiliser decompositions which scale with the number of non-Clifford gates in a circuit. By using magic state injection, we can write a state $|\psi\rangle$ obtained from a Clifford+T circuit as a Clifford circuit that takes as input a T magic state $|T\rangle := \frac{|0\rangle + e^{i\frac{\pi}{4}}|1\rangle}{\sqrt{2}}$ for each T gate in the original circuit. One can then decompose these T magic states, and hence $|\psi\rangle$ into a sum of stabiliser states. Since each of the individual terms can be simulated efficiently, the simulation cost of the state scales with the number of terms in the decomposition.

Naïvely, one can decompose each of the T magic states individually as a linear combination of the stabiliser states $|0\rangle$ and $|1\rangle$, obtaining 2^t terms. However, by decomposing larger non-stabiliser states at once, one can obtain lower simulation costs [7, 16]. Until now, such techniques have focused on decomposing many identical copies of a fixed magic state. For example, much work has gone into finding efficient decompositions of $|T\rangle^{\otimes t}$ for different values of t , with the best known decomposition scaling asymptotically as $2^{\alpha t}$ with $\alpha \approx 0.396$ [16].

In previous work by some of the current authors [15], a new method for simulation with stabiliser decompositions was introduced, which works by representing the circuit to be simulated as a *ZX-diagram* [8, 9], and then interleaving the decompositions of [7] with the diagram simplification strategy of [14] to reduce the number of stabiliser terms required, sometimes by many orders of magnitude. The scheme goes as follows:

1. Translate the circuit, together with the desired input state and measurement effect, to a ZX-diagram.
2. Simplify the diagram as much as possible using the rules of the ZX-calculus.
3. Pick a set of non-Clifford generators and decompose them, obtaining a weighted sum of diagrams with fewer non-Clifford generators.
4. Apply the previous two steps recursively to each of the diagrams until no non-Clifford generators remain, in which case each diagram is simplified to a single complex number.
5. The sum of these numbers gives the overall amplitude.

This allowed for the simulation of significantly larger circuits than before. While the previous best simulated a 50-qubit circuit with 64 T gates, in Ref. [15] they simulated 50-qubit circuits with up to 1400 T gates.

In this article we build on the work of both [16] and [15], by significantly improving Step 3. Our main contribution is to show that it is useful to consider different states than just $|T\rangle^{\otimes t}$ for decomposition. This leads to two improvements. The first is that if certain entangled states are present in the ZX-diagram to be simulated, then we can use a decomposition that asymptotically requires significantly fewer terms, having exponential parameter $\alpha \approx 0.25$ instead of $\alpha \approx 0.396$. While these special states need not appear in a generic ZX-diagram obtained from a Clifford+T circuit, our empirical data suggests that they often will. It is crucial here that we represent the intermediate stages of the simulation as a ZX-diagram, as opposed to a circuit with magic states as input, as the graph structure makes it possible to find the right entangled states to be decomposed.

Our second improvement is that one of these special decompositions can be adapted to lead to a decomposition of $|T\rangle^{\otimes 5}$ that uses only 3 terms but leaves one $|T\rangle$ in the resulting reduced terms, which effectively gives us a “4-to-3” decomposition meaning we require $2^{\alpha t}$

terms with $\alpha \approx 0.396$ in the worst case. This matches the asymptotic upper bound reported in Ref. [16], however unlike in their method, we obtain this bound at a fixed finite size rather than in the limit.

We implemented our new decompositions in `quixZ`, the software used in Ref. [15], and assessed the performance of our method by benchmarking the same type of quantum circuits. We found that our new scheme always outperforms the previous one, often by several orders of magnitude. For instance, we benchmarked 125 different 20-qubit hidden-shift circuits with T-count 112. While our method used at most 1 second for each of these, the method of [15] required more than 1000 seconds for some. We also found the distribution of the required time to be a lot less erratic: while 17% of random hidden shift circuits on 50 qubits with T-count 1400 could be sampled within 5 minutes in the previous proposal, 98% pass the test with the new decompositions – and the remaining 2% only require 1 additional minute, meaning we can reliably simulate 50-qubit hidden-shift circuits with over 1000 T gates on a consumer laptop.

In Section 2 we present the ZX-calculus, the formalism used throughout the rest of the paper, and in Section 3 we discuss the state-of-the-art in stabiliser rank and stabiliser decompositions. In Section 4 we introduce the decompositions of the entangled states that we will use, as well as a “partial” stabiliser decomposition of $|T\rangle^{\otimes 5}$. Finally, in Section 5, we conduct some benchmarks to assess the relevance of the new decompositions. We end with some concluding remarks in Section 6.

2 ZX-calculus

The ZX-calculus is a graphical language for reasoning about quantum computation [8, 9]. It represents quantum processes using ZX-diagrams which can then graphically be rewritten and simplified using a collection of rewrite rules. For a review see [18]. Here we give a brief overview of the notions we will need.

ZX-diagrams [8] are built from a set of generators: *Z-spiders* represented by green dots, *X-spiders* represented by red dots, the Hadamard gate represented by a yellow box, and generators capturing “half-turns” of a wire and wire crossings. These are defined as follows:

$$\begin{array}{l}
 \begin{array}{c} \vdots \\ \vdots \\ \text{---} \alpha \text{---} \\ \vdots \\ \vdots \end{array} = |0\dots 0\rangle\langle 0\dots 0| + e^{i\alpha}|1\dots 1\rangle\langle 1\dots 1| \\
 \begin{array}{c} \vdots \\ \vdots \\ \text{---} \alpha \text{---} \\ \vdots \\ \vdots \end{array} = |+\dots+\rangle\langle +\dots+| + e^{i\alpha}|-\dots-\rangle\langle -\dots-| \\
 \text{---} \square \text{---} = |+\rangle\langle 0| + |-\rangle\langle 1|
 \end{array}
 \qquad
 \begin{array}{l}
 \text{---} = |0\rangle\langle 0| + |1\rangle\langle 1| \\
 \text{---} \times \text{---} = \sum_{i,j \in \{0,1\}} |ij\rangle\langle ji| \\
 (= |00\rangle + |11\rangle \\
) = \langle 00| + \langle 11|
 \end{array}$$

Note that, much like in quantum circuit notation, we interpret composition as “plugging” diagrams together and tensor product as putting diagrams side-by-side:

$$\left(\begin{array}{c} \vdots \\ \vdots \\ \square_{D_2} \\ \vdots \\ \vdots \end{array} \right) \circ \left(\begin{array}{c} \vdots \\ \vdots \\ \square_{D_1} \\ \vdots \\ \vdots \end{array} \right) = \begin{array}{c} \vdots \\ \vdots \\ \square_{D_1} \square_{D_2} \\ \vdots \\ \vdots \end{array}
 \qquad
 \left(\begin{array}{c} \vdots \\ \vdots \\ \square_{D_1} \\ \vdots \\ \vdots \end{array} \right) \otimes \left(\begin{array}{c} \vdots \\ \vdots \\ \square_{D_2} \\ \vdots \\ \vdots \end{array} \right) = \begin{array}{c} \vdots \\ \vdots \\ \square_{D_1} \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ \square_{D_2} \\ \vdots \\ \vdots \end{array}$$

5:4 Classical Simulation with Partial and Graphical Stabiliser Decompositions

We can hence build complicated diagrams by composition of these small bricks, and interpret them as a (not necessarily unitary) linear map. For example, every quantum circuit built from the traditional gate set $\langle \text{CNOT}, H, Z_\alpha \rangle$ can be directly mapped into a ZX-diagram via the translation:

$$\text{CNOT} \mapsto \sqrt{2} \begin{array}{c} \circlearrowleft \\ \text{---} \\ \circlearrowright \end{array} \quad Z_\alpha \mapsto \text{---} \circlearrowleft \alpha \text{---} \quad H \mapsto \text{---} \square \text{---}$$

Notice that we used here a complex number explicitly as a global multiplicative scalar to the CNOT diagram. These scalars can be represented in ZX, but are a lot more convenient to deal with explicitly when possible. Any diagram with 0 inputs and 0 outputs represents a scalar, and when it can be efficiently computed, we will simply do this, instead of leaving it as a diagram.

An important subclass of ZX-diagrams are the *Clifford ZX-diagrams*, i.e. those whose spiders only have phases equal to integer multiples of $\pi/2$. A Clifford ZX-diagram with no input wires always represents a stabiliser state, and conversely any stabiliser state can always be written as a Clifford ZX-diagram [3].

A convenient feature of ZX-diagrams is that the linear map they represent only depends on the connectivity of the diagram and not the actual positions of spiders or the direction of wires between them. This means in particular that we can topologically deform a diagram however we wish without changing its interpretation. This allows us to treat a ZX-diagram as an undirected open (multi-)graph, whose vertices are Z-spiders, X-spiders and H-gates. As H-gates are binary, it can be convenient to internalise them on their respective wires, and consider a new wire type which we call an *H-edge*, represented by a dashed blue line:

$$\begin{array}{c} \diagup \quad \diagdown \\ \circlearrowleft \text{---} \quad \text{---} \quad \circlearrowleft \\ \diagdown \quad \diagup \end{array} \text{---} \text{---} \text{---} \quad := \quad \begin{array}{c} \diagup \quad \diagdown \\ \circlearrowleft \text{---} \quad \square \quad \text{---} \quad \circlearrowleft \\ \diagdown \quad \diagup \end{array}$$

In addition to topological deformation, there are graphical rewrite rules that also preserve the semantics of the diagram, and in fact, *complete* axiomatisations of the ZX-calculus exist (e.g. [12, 13, 19]), which entirely capture the semantical equivalence of diagrams. We will not give a full such axiomatisation here, but only those rules we will need in this paper:

$$\begin{array}{c} \circlearrowleft \alpha \\ \vdots \\ \circlearrowleft \beta \\ \vdots \end{array} = \begin{array}{c} \circlearrowleft \alpha + \beta \\ \vdots \end{array}, \quad \square \square = \text{---}, \quad \circlearrowleft = \text{---}, \quad \begin{array}{c} \square \\ \square \\ \circlearrowleft \alpha \\ \square \\ \square \end{array} = \begin{array}{c} \square \\ \square \\ \circlearrowright \alpha \\ \square \\ \square \end{array}$$

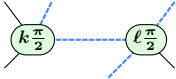
Note that these rules remain true when Z- and X-spiders are interchanged (i.e. when the colours are swapped).

Using these rules, it is possible to turn any diagram into a form where i) all spiders are Z-spiders and ii) all edges are H-edges (but inputs and outputs remain plain wires). A diagram in such a form is called *graph-like* [11]. These are the diagrams we will work with in this paper.



While the rewrite rules are unidirectional, we can apply one-way rewrite strategies that try to reduce some metric of the diagram, such as the number of spiders in a diagram. In particular, in Ref. [11] a simplification strategy was introduced that can remove most Clifford spiders, i.e. those spiders whose phase is an integer multiple of $\pi/2$, from a ZX-diagram. It is this rewrite strategy (or more specifically, the improved version from Ref. [14]) that was used in Ref. [15] to simplify the diagrams necessary for stabiliser decomposition-based simulation. As the details are not relevant to us, we will only describe some of the properties

that the resulting diagrams enjoy, as these will be important for our results. We will call diagrams simplified by the rewrite strategy of Ref. [14] *reduced*. Note that a spider is called *internal* if it is not directly connected to an input or an output of the diagram.

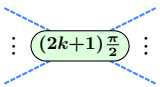
► **Lemma 1.** *In reduced diagrams, no two Clifford spiders are neighbours.*

In other words, patterns like  won't appear in the diagram.

► **Lemma 2.** *In reduced diagrams, no internal Clifford spider can have arity ≤ 2 .*

In other words, patterns like  and  won't appear.


► **Lemma 3.** *In reduced diagrams, no internal spider phases are odd multiples of $\frac{\pi}{2}$.*

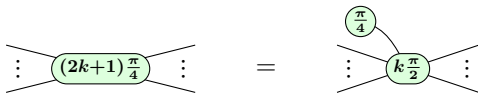
In other words, patterns like  won't appear.

3 Stabiliser Rank and stabiliser decompositions

Since stabiliser states/operators are efficiently classically simulable, it is worthwhile to try to find decompositions of arbitrary quantum states as linear combination of stabiliser states with a small number of terms.

More formally, let $|\psi\rangle$ be an arbitrary state. Then a *stabiliser decomposition* of $|\psi\rangle$ is a decomposition $|\psi\rangle = \sum_k^n \lambda_k |\psi_k\rangle$ where the $|\psi_k\rangle$ are all stabiliser states and the $\lambda_k \in \mathbb{C}$ are some scalars. The smallest n for which such a decomposition exists is called the stabiliser rank of $|\psi\rangle$, and is denoted by $\chi(|\psi\rangle)$. For simulation purposes, we are obviously not only interested in $\chi(|\psi\rangle)$ but also in an associated decomposition.

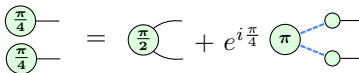
In the Clifford+T case, most of the results on stabiliser decompositions have focussed on tensor products of the magic state $|T\rangle := \frac{1}{\sqrt{2}}(|0\rangle + e^{i\frac{\pi}{4}}|1\rangle)$, which we can represent as a ZX-diagram (up to normalisation) by . Whenever we have a spider with phase an odd multiple of $\pi/4$ we can unfuse a magic state:



A single magic state can be decomposed as a sum of two stabiliser states:

$$\text{Diagram of } \pi/4 \text{ spider} = \frac{1}{\sqrt{2}} (\text{Diagram of } \pi/2 \text{ spider} + e^{i\pi/4} \text{Diagram of } \pi \text{ spider})$$

A naive decomposition of $\text{Diagram of } \pi/4 \text{ spider}^{\otimes t}$ would then be to apply this decomposition for all magic states, which would result in a decomposition with 2^t terms. More sophisticated decompositions exist however, such as:



We can use this decomposition to reduce T magic states pairwise, bringing the total number of terms down to $2^{\frac{t}{2}}$. There is also a $|T\rangle^{\otimes 6}$ decomposition requiring 7 terms, described in [7] and used in [15]. More recently, an improved decomposition of $|T\rangle^{\otimes 6}$ was found that only requires 6 terms [16]. To compare different decompositions it will be helpful to consider how

many stabiliser terms would be needed if we were to decompose every non-Clifford phase in the diagram using that decomposition. If a decomposition reduces the T-count by r using p terms, we would need $p^{t/r} = 2^{\alpha t}$ stabiliser terms. This $\alpha = \log_2(p)/r$ will be the metric we want to minimise. For instance, the decomposition of [7] gives $\alpha \approx 0.468$ ($r = 6, p = 7$).

In the following, we will consider some entangled non-stabiliser states of which we can find a better decomposition than would be suggested by the number of magic states needed to write down these states.

4 Results

We present two different improvements to the stabiliser decompositions used before: using decompositions of certain entangled states instead of product states, and using “partial” stabiliser decompositions where the terms are not Clifford, but merely contain a lower amount of magic states.

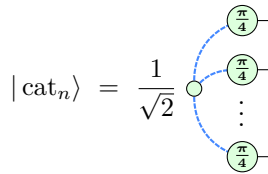
4.1 Cat states in the ZX-calculus

The states we are interested in here were introduced in Ref. [16] as *cat states*:

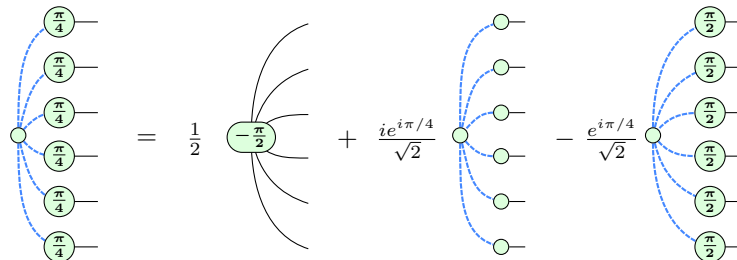
$$|\text{cat}_n\rangle := \frac{1}{\sqrt{2}}(\mathbb{I}^{\otimes n} + Z^{\otimes n})|T\rangle^{\otimes n} = \frac{1}{\sqrt{2^{n+1}}} \left(\left(\frac{\pi}{4}\right)^{\otimes n} + \left(\frac{5\pi}{4}\right)^{\otimes n} \right)$$

In Ref. [16] these states are used because i) they have a relatively small stabiliser rank and ii) because if the stabiliser rank of $|\text{cat}_n\rangle$ is bounded by c , then that of $|T\rangle^{\otimes n}$ is bounded by $2c$. For instance, they find a decomposition of $|\text{cat}_6\rangle$ using 3 stabiliser states, which hence gives a decomposition of $|T\rangle^{\otimes 6}$ using 6 stabiliser states. This improves upon the 6-to-7 decomposition of Ref. [7], and improves $\alpha \approx 0.468$ to $\alpha \approx 0.431$.

In this paper we focus on the use of the cat-states directly for stabiliser decompositions, rather than first transforming them into decompositions of sets of T states. First, notice that they can be easily expressed as ZX-diagrams:

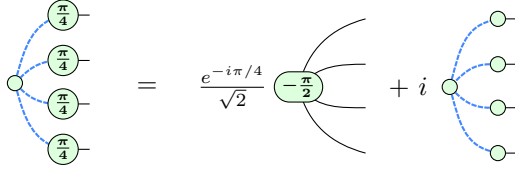


We can then translate to ZX the decomposition of $|\text{cat}_6\rangle$ found in Ref. [16]¹:



¹ The description of the last diagram in the decomposition is not the one given in [16]. However, the two can be shown to be equivalent using the ZX-calculus.

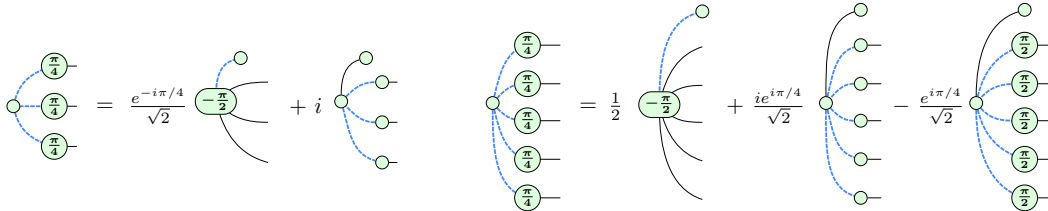
Hence, if our diagram contained enough subdiagrams that look like $|\text{cat}_6\rangle$ to allow us to decompose all non-Clifford spiders using this decomposition we would get $3^{\frac{t}{6}} = 2^{\alpha t}$ terms, where $\alpha \approx 0.264$. We can in fact do even better by considering the decomposition of $|\text{cat}_4\rangle$, which has a stabiliser rank of only 2:



If we could use only this decomposition we would get $2^{0.25t}$ terms.

The existence of these beneficial decompositions suggests a new strategy: at every step of the decomposition of the complete diagram, we look at every internal spider in the graph and the shape of its neighbourhood and we pick a target for decomposition that requires the fewest number of terms per magic state. After each decomposition, we use a ZX-diagram simplification strategy (like the one in Ref. [15]) and then we repeat the procedure.

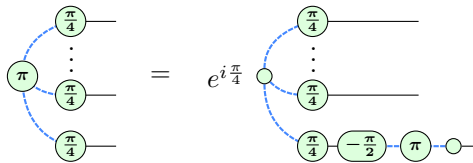
We cannot hope to always have an occurrence of $|\text{cat}_4\rangle$ or $|\text{cat}_6\rangle$ at hand. However, notice that we can always infer a decomposition of $|\text{cat}_n\rangle$ from one of $|\text{cat}_{n+k}\rangle$ ($k > 0$), by $|\text{cat}_n\rangle = \langle 0|^{\otimes k} \otimes \mathbb{I}^{\otimes n} |\text{cat}_{n+k}\rangle$. This observation gives us for instance a decomposition of $|\text{cat}_3\rangle$ and $|\text{cat}_5\rangle$:



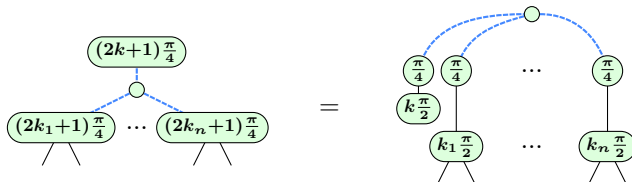
These two decompositions give us respectively $\alpha = 1/3 \approx 0.333$ and $\alpha \approx 0.317$.

Note that by Lemma 2, there will never be any occurrence of $|\text{cat}_2\rangle$ or $|\text{cat}_1\rangle$ in our simplified diagrams so that we now have a strategy for decomposing any $|\text{cat}_n\rangle$ for $n \leq 6$.

These decompositions apply when the “central” Clifford spider has phase 0. By Lemma 3 any internal Clifford spider will have phase 0 or π . The π case is easily taken care of by pushing the phase through one of the neighbours as follows:



Note that in practice, in our reduced diagrams the only occurrences of cat-states will be as *phase gadgets* [14, 10], which relate to cat states in the following way:

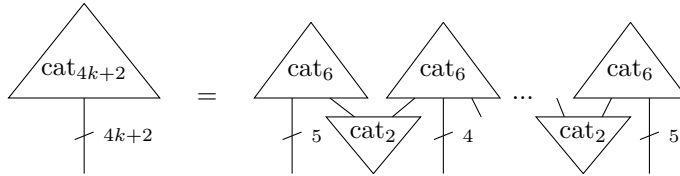


Hence, an n -legged phase gadget corresponds to a $|\text{cat}_{n+1}\rangle$ state, so that it can be decomposed more efficiently.

In Ref. [16] they also describe ways to get efficient stabiliser decompositions of $|\text{cat}_n\rangle$ for bigger n . For instance, they find a decomposition of $|\text{cat}_{10}\rangle$ which has 9 terms, which gives $\alpha \approx 0.317$. For simplicity of implementation, we don't consider these decompositions of bigger states here.

4.2 Partial magic state decompositions

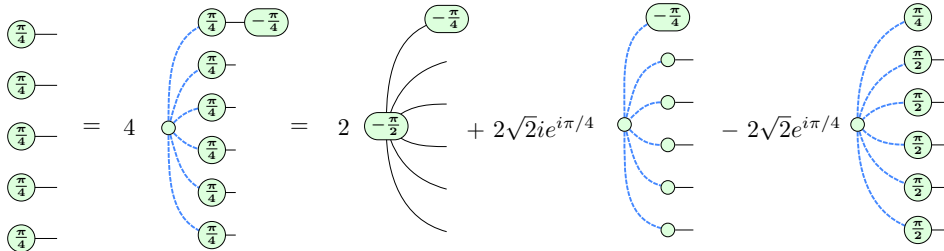
If we cannot find cat states in our diagram (which means that it only has non-Clifford spiders or that every Clifford spider has arity larger than 6), then we could fall back to the magic state decompositions described in Ref. [16]. Namely, as is observed in Ref. [16], we can construct decompositions of bigger cat-states by the following identity:



Keep in mind that $|\text{cat}_n\rangle = (|0\rangle^{\otimes k} \otimes |\mathbb{I}^{\otimes n}\rangle) |\text{cat}_{n+k}\rangle$ so that we also get decompositions for n that are not equal to $4k + 2$. Using this construction of $|\text{cat}_{4k+2}\rangle$ we get a decomposition of $|T\rangle^{\otimes 4k+2}$ that uses $2 \cdot 3^k$ terms. This decomposition requires $2^{\alpha(4k+2)}$ terms where

$$\alpha = \frac{1 + k \log_2(3)}{4k + 2} \xrightarrow{k \rightarrow \infty} \frac{\log_2(3)}{4} \approx 0.396.$$

But as we will see now, we can in fact reach this asymptotic α without needing to decompose all magic states at once. The idea here is to use the decomposition of $|\text{cat}_6\rangle$ to reduce the T-count of the diagram by 4, using only 3 terms. This can be done whenever we have a T-count ≥ 5 , in the following way:



This is a (non-stabiliser) decomposition of the 5-qubit magic state which leaves one T-spider in each term. So each term effectively loses 4 magic states, so that we can view this as a 4-to-3 strategy. We get for this operation $\alpha = \frac{\log_2(3)}{4} \approx 0.396$ so that we reach this number concretely that in Ref. [16] was only reached asymptotically.

Additional advantages of this decomposition over the strategy using the $|\text{cat}_{4k+2}\rangle$ decomposition, are that i) it is simpler to handle and implement (as a smaller part of the diagram is involved) and ii) it more easily allows for the ZX-diagram reduction strategy of [15] to be used in-between rounds of decompositions, so that there is more possibility for reduction of the number of terms.

We remark that this type of “partial decomposition” can be used to decompose any $|T\rangle^{\otimes 4k+1}$ state using the $|\text{cat}_{4k+2}\rangle$ decomposition, but doing so always results in the same α , whenever $k \geq 1$, so that there is no benefit to doing so (at least from an asymptotic perspective).

4.3 The full decomposition strategy

The conclusion of the previous sections is that some subdiagrams are better to decompose into stabilisers than others. In particular, in our diagrams we should look, in order of importance, for:

1. a Clifford spider with 4 neighbours ($\alpha = 0.25$),
2. a Clifford spider with 6 neighbours ($\alpha \approx 0.264$),
3. a Clifford spider with 5 neighbours ($\alpha \approx 0.317$),
4. a Clifford spider with 3 neighbours ($\alpha = 1/3 \approx 0.333$),
5. any 5 T-spiders ($\alpha \approx 0.396$).

Once the best subdiagram to decompose has been found, we apply the associated decomposition, then simplify each resulting diagram using the optimisation strategy of [14] as described in [15], and repeat the procedure for each term until we are left with diagrams that have fewer than 5 T-spiders and no cat-states, in which case we simply fall back to the usual magic-state decompositions.

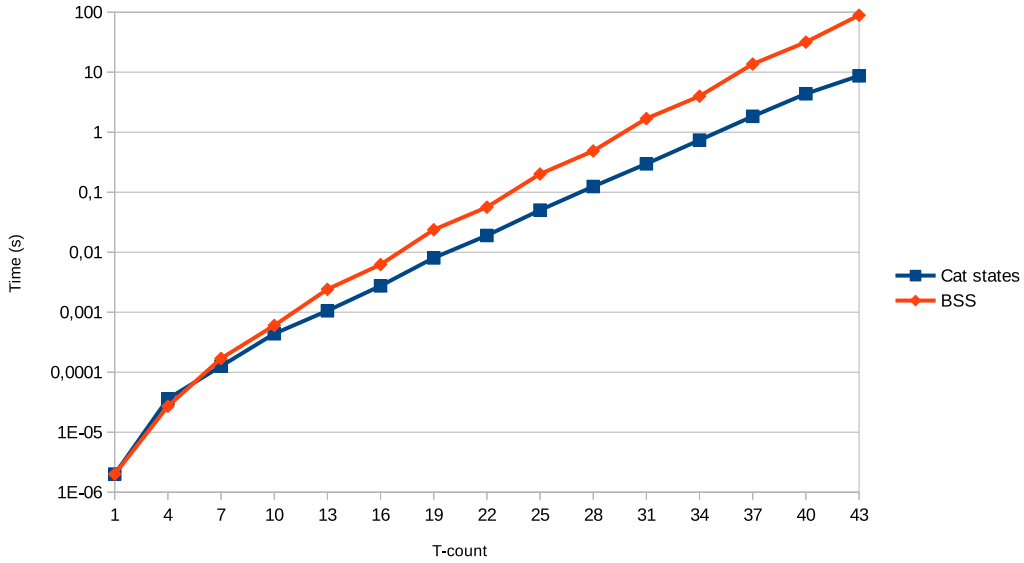
5 Benchmarking

To assess the effectiveness of our approach, we performed simulations of the two families of circuits that were benchmarked in Ref. [15], and compare the associated running times with theirs. For a better comparison, we turned off parallelisation in most experiments. All subsequent tests were performed on a consumer laptop, equipped with an 2.60GHz Intel Core i5-10400H CPU. By using a depth-first decomposition approach the amount of memory needed for these benchmarks was insignificant. In these benchmarks we refer to the approach of Ref. [15] as *BSS*, which stands for Bravyi, Smith, Smolin, as it uses the decomposition introduced in Ref. [7].

5.1 Random Clifford+T circuits

The first family of circuits we consider are Clifford+T circuits. We construct these by composing a given number (the T-count t) of *exponentiated Pauli unitaries*, which are of the form $\exp(-i(2k+1)\frac{\pi}{4}P)$ for P some Pauli string, i.e. a tensor product of 1-qubit Pauli operators. The *weights* of the Pauli strings (the number of non-identity operators) is chosen randomly between 2 and 4, in order to mimic the structure common to quantum chemistry circuits, where the Hamiltonian has terms of weight 2–4.

We considered random 20-qubit Clifford+T circuits with T-counts varying from 1 to 43 (increasing in steps of 3). We calculated the amplitude of these circuits for a fixed input of $|+\rangle^{\otimes 20}$ and output of $\langle +|^{\otimes 20}$ (where $|+\rangle := \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$) using the two decomposition strategies. The running times shown in Figure 1 are averages over 10 runs for each T-count. We see already an order of magnitude improvement between the running time of the two approaches for a T-count of 43.



■ **Figure 1** Runtime of random 20-qubit Clifford+T circuit simulations (avg of 10 runs per T-count).

5.2 Random hidden-shift circuits

The second family of circuits we consider are hidden-shift circuits, whose description can be found in Refs. [4, 5]. These circuits are composed of H, Z, CZ and CCZ gates. The first three are readily translatable to ZX-diagrams, while the CCZ gate requires a decomposition that introduces T-spiders. The decomposition used here is one that uses 7 T-spiders for each CCZ gate.²

We found the cat-state decompositions to be particularly efficient compared to the BSS decomposition for this class of circuits. We empirically found the BSS-based approach to be quite erratic, with running times varying between 0.005s and more than 1000s for circuits with the same T-count. This aspect seems to be dampened when using cat-state decompositions. To better gauge this phenomenon, we generated 125 20-qubit hidden-shift circuits with T-count 112, and simulated all the circuits using both approaches. The results, sorted by the simulation times for the cat-state method, can be seen in Figure 2. In the most extreme case, for simulating the same circuit, the cat-state-based decomposition ran in less than half a second, while the BSS-based one took more than 1 h 25 m.

The distribution of running times for both approaches can be found in Figure 3. As we are dealing with an inherently exponential process, it makes sense that perturbations have an exponential effect on the result. This is why we plotted the distributions on a logarithmic scale. In this same scale, it is possible to compute the variance of both distributions. We get a variance of $\sigma^2 \simeq 0.523$ for the cat-state approach, and a variance of $\sigma^2 \simeq 3.02$ for the BSS approach. This is strong evidence that the erratic behaviour is indeed dampened with our decomposition scheme, even when taking the exponential scaling into account.

This erratic behaviour being lifted with the cat-state strategy, we were able to run tests without the need for a time limit, which Ref. [15] set at 5 minutes. We re-ran the experiment that performed strong simulation of 100 random 50-qubit hidden-shift circuits with T-count

² As discussed in Ref. [15], the 4 T-count decomposition of a CCZ gate that uses an ancilla turns out to work less well in practice, as it allows the diagrammatic simplifier to find fewer optimisations.

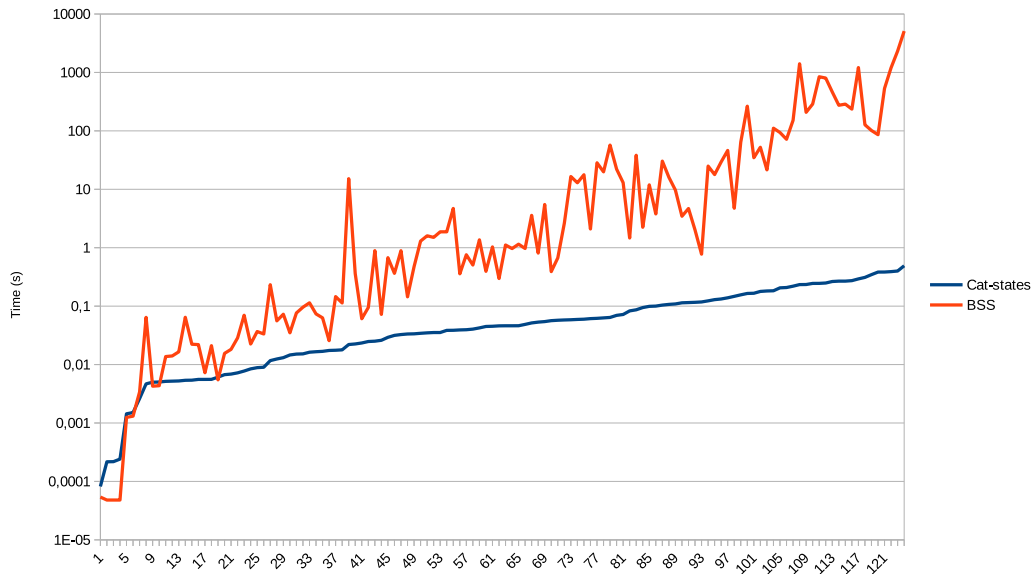


Figure 2 Logarithmic plot of the runtimes of simulating 125 different 20-qubit hidden-shift circuits with T-count 112, using the BSS decomposition and our strategy. Each datapoint corresponds to a circuit and they are sorted left-to-right according to simulation time using our approach.

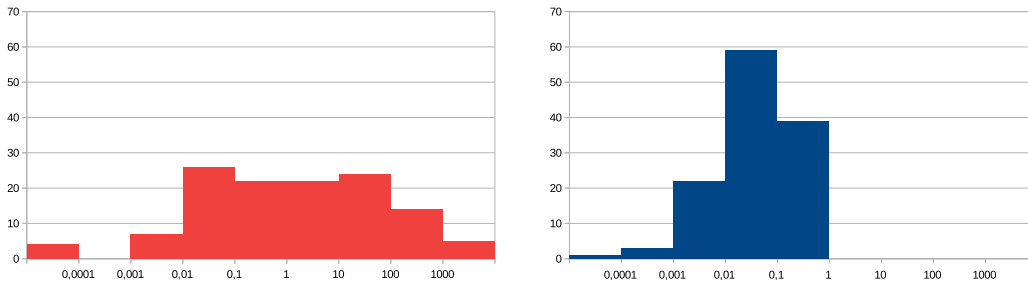
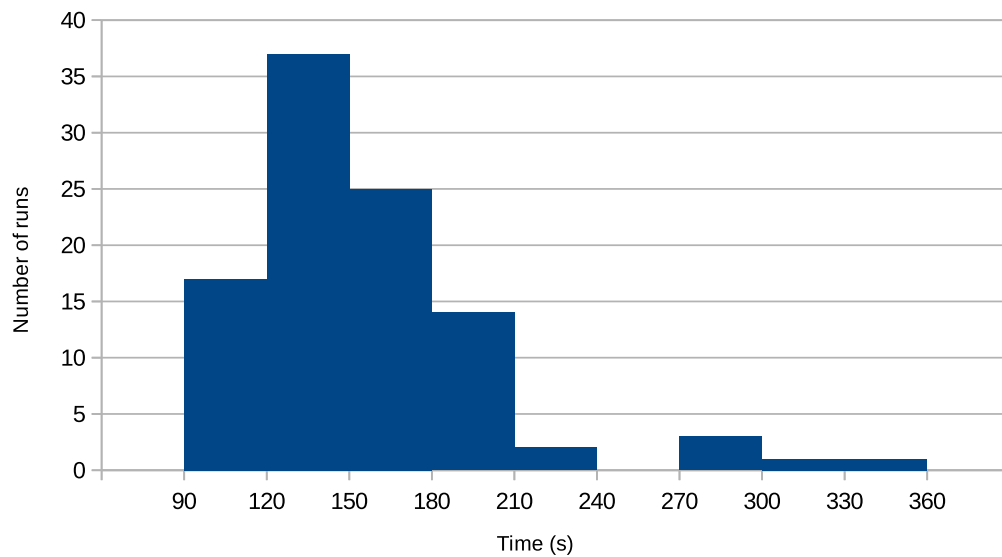


Figure 3 Distributions of running times (in seconds) of simulating 125 different 20-qubit hidden-shift circuits with T-count 112 using the BSS decomposition (left) and the cat-state decomposition (right).

1400 using our new decompositions. We observed that 98% of the runs took less than 5 minutes, with the remaining circuits finishing within 6 minutes. This is to be compared with the BSS decomposition approach which in [15] obtained a 17% success rate (with the 5 minutes limit) even though it ran on a dedicated server with better CPU and parallelisation. The running time distribution is given in Figure 4.

6 Conclusion and Further Work

In this paper we built on the stabiliser rank quantum simulation method by combining it with a ZX-calculus simplification strategy and novel decompositions of entangled non-stabiliser states. We additionally introduced a new technique of *partial* magic state decompositions that only reduce the T-count instead of eliminating all magic states at once, which achieves the stabiliser rank upper bound of $2^{0.396t}$ found in Ref. [16] in a direct way instead of just



■ **Figure 4** The time distribution of simulating 100 random 50-qubit hidden-shift circuits with T-count 1400 using our new decompositions.

asymptotically. Our benchmarks shows that our technique is in the best case several orders of magnitude faster than the strategy of Ref. [15], which itself was already capable of simulating much larger circuits than the previous best. Additionally, our techniques seem to “smooth out” the erratic runtimes of Ref. [15] allowing for more consistent simulation times.

For future work it would be interesting to investigate whether other classes of entangled states with an even smaller stabiliser rank could be found and used for this kind of exact simulation. It is also worth investigating how decompositions of these states can be used in methods for *approximate* simulations, like the stabiliser extent methods, or the norm estimation technique used in Ref. [4].

To sample from the random Clifford+T circuits we used the same technique as Ref. [15], which required a technique that doubles the T-count in the worst case. It would be interesting to see if the recent technique of calculating marginals without doubling [6] would help improve run times.

References

- 1 Scott Aaronson and Daniel Gottesman. Improved simulation of stabilizer circuits. *Physical Review A*, 70(5):052328, 2004. doi:10.1103/PhysRevA.70.052328.
- 2 Dorit Aharonov. A simple proof that Toffoli and Hadamard are quantum universal. *arXiv preprint*, 2003. arXiv:quant-ph/0301040.
- 3 Miriam Backens. The ZX-calculus is complete for stabilizer quantum mechanics. *New Journal of Physics*, 16(9):093021, 2014. doi:10.1088/1367-2630/16/9/093021.
- 4 Sergey Bravyi, Dan Browne, Pdraic Calpin, Earl Campbell, David Gosset, and Mark Howard. Simulation of quantum circuits by low-rank stabilizer decompositions. *Quantum*, 3:181, September 2019. doi:10.22331/q-2019-09-02-181.

- 5 Sergey Bravyi and David Gosset. Improved classical simulation of quantum circuits dominated by Clifford gates. *Physical Review Letters*, 116(25), June 2016. doi:10.1103/PhysRevLett.116.250501.
- 6 Sergey Bravyi, David Gosset, and Yunchen Liu. How to simulate quantum measurement without computing marginals. *arXiv preprint*, 2021. arXiv:2112.08499.
- 7 Sergey Bravyi, Graeme Smith, and John A Smolin. Trading classical and quantum computational resources. *Physical Review X*, 6(2):021043, 2016. doi:10.1103/PhysRevX.6.021043.
- 8 Bob Coecke and Ross Duncan. Interacting quantum observables. In *Proceedings of the 37th International Colloquium on Automata, Languages and Programming (ICALP)*, Lecture Notes in Computer Science, 2008. doi:10.1007/978-3-540-70583-3_25.
- 9 Bob Coecke and Ross Duncan. Interacting quantum observables: categorical algebra and diagrammatics. *New Journal of Physics*, 13:043016, 2011. doi:10.1088/1367-2630/13/4/043016.
- 10 Alexander Cowtan, Silas Dilkes, Ross Duncan, Will Simmons, and Seyon Sivarajah. Phase Gadget Synthesis for Shallow Circuits. In Bob Coecke and Matthew Leifer, editors, *Proceedings 16th International Conference on Quantum Physics and Logic, Chapman University, Orange, CA, USA., 10-14 June 2019*, volume 318 of *Electronic Proceedings in Theoretical Computer Science*, pages 213–228. Open Publishing Association, 2020. doi:10.4204/EPTCS.318.13.
- 11 Ross Duncan, Aleks Kissinger, Simon Perdrix, and John van de Wetering. Graph-theoretic Simplification of Quantum Circuits with the ZX-calculus. *Quantum*, 4:279, June 2020. doi:10.22331/q-2020-06-04-279.
- 12 Amar Hadzihasanovic, Kang Feng Ng, and Quanlong Wang. Two complete axiomatisations of pure-state qubit quantum computing. In *Proceedings of the 33rd Annual ACM/IEEE Symposium on Logic in Computer Science, LICS '18*, pages 502–511, New York, NY, USA, 2018. ACM. doi:10.1145/3209108.3209128.
- 13 Emmanuel Jeandel, Simon Perdrix, and Renaud Vilmart. A Complete Axiomatisation of the ZX-calculus for Clifford+T Quantum Mechanics. In *Proceedings of the 33rd Annual ACM/IEEE Symposium on Logic in Computer Science*, pages 559–568. ACM, 2018. doi:10.1145/3209108.3209131.
- 14 Aleks Kissinger and John van de Wetering. Reducing the number of non-Clifford gates in quantum circuits. *Physical Review A*, 102:022406, August 2020. doi:10.1103/PhysRevA.102.022406.
- 15 Aleks Kissinger and John van de Wetering. Simulating quantum circuits with ZX-calculus reduced stabiliser decompositions. *Quantum Science and Technology*, 2022. doi:10.1088/2058-9565/ac5d20.
- 16 Hammam Qassim, Hakop Pashayan, and David Gosset. Improved upper bounds on the stabilizer rank of magic states. *Quantum*, 5:606, December 2021. doi:10.22331/q-2021-12-20-606.
- 17 Yaoyun Shi. Both Toffoli and controlled-NOT need little help to do universal quantum computation. Preprint, 2002. arXiv:quant-ph/0205115.
- 18 John van de Wetering. ZX-calculus for the working quantum computer scientist. Preprint, 2020. arXiv:2012.13966.
- 19 Renaud Vilmart. A Near-Minimal Axiomatisation of ZX-Calculus for Pure Qubit Quantum Mechanics. In *2019 34th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)*, pages 1–10, 2019. doi:10.1109/LICS.2019.8785765.

On Converses to the Polynomial Method

Jop Briët  

CWI & QuSoft, Amsterdam, The Netherlands

Francisco Escudero Gutiérrez 

CWI & QuSoft, Amsterdam, The Netherlands

Abstract

A surprising “converse to the polynomial method” of Aaronson et al. (CCC’16) shows that any bounded quadratic polynomial can be computed exactly in expectation by a 1-query algorithm up to a universal multiplicative factor related to the famous Grothendieck constant. A natural question posed there asks if bounded quartic polynomials can be approximated by 2-query quantum algorithms. Arunachalam, Palazuelos and the first author showed that there is no direct analogue of the result of Aaronson et al. in this case. We improve on this result in the following ways: First, we point out and fix a small error in the construction that has to do with a translation from cubic to quartic polynomials. Second, we give a completely explicit example based on techniques from additive combinatorics. Third, we show that the result still holds when we allow for a small additive error. For this, we apply an SDP characterization of Gribling and Laurent (QIP’19) for the completely-bounded approximate degree.

2012 ACM Subject Classification Mathematics of computing → Functional analysis; Theory of computation → Quantum complexity theory

Keywords and phrases Quantum query complexity, polynomial method, completely bounded polynomials

Digital Object Identifier 10.4230/LIPIcs.TQC.2022.6

Related Version *Full Version:* <https://arxiv.org/abs/2204.12303>

Funding *Jop Briët:* This research was supported by the European Union’s Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement no. 945045, and by the NWO Gravitation project NETWORKS under grant no. 024.002.003.

Francisco Escudero Gutiérrez: This research was supported by the European Union’s Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement no. 945045, and by the NWO Gravitation project NETWORKS under grant no. 024.002.003.

Acknowledgements We want to thank Srinivasan Arunachalam, Sander Gribling and Carlos Palazuelos for useful discussions. We also want to thank the referees of TQC for their helpful comments.

1 Introduction

A celebrated result of Beals et al. [6], known as the *polynomial method* in quantum complexity theory, leverages the problem of lower bounding the quantum query complexity of a Boolean function to lower bounding the approximate degree. The method is based on the fact that for every t -query quantum algorithm \mathcal{A} that takes an n -bit input and returns a sign, there is a real n -variable polynomial f of degree at most $2t$ such that $f(x) = \mathbb{E}[\mathcal{A}(x)]$ for every x . Here, the expectation is taken with respect to the randomness in the measurement done by \mathcal{A} .¹ In addition to many new lower bounds, this result led to a line of research on possible

¹ We identify a quantum query algorithm with the (random) function representing its output on a given input string.



converses, whereby a bounded polynomial f can be turned into a quantum query algorithm that approximates f and whose query complexity depends in some reasonably way on the degree of f . Here, f is *bounded* if it maps the Boolean hypercube to the interval $[-1, 1]$ and a quantum query algorithm \mathcal{A} *approximates* f if for some constant error parameter $\varepsilon < 1$, we have that $|f(x) - \mathbb{E}[\mathcal{A}(x)]| \leq \varepsilon$ for every x . For bounded polynomials of degree at most 2, the following converse was proved in [2], using a surprising application of the Grothendieck inequality from Banach space theory (we refer to [14] for an extensive survey on Grothendieck-type inequalities).

► **Theorem 1** (Aaronson et al.). *There exists an absolute constant $C > 0$ such that the following holds. For every bounded polynomial f of degree at most 2, there exists a one-query quantum algorithm \mathcal{A} such that $\mathbb{E}[\mathcal{A}(x)] = Cf(x)$ holds for every $x \in \{-1, 1\}^n$.*

This “multiplicative converse” implies an approximation with *additive* error at most $1 - C$. A natural question is if this result generalizes to quartic polynomials and two-query quantum algorithms [2, Section 5, Question 1]. Based on the probabilistic method and a new characterization of quantum query algorithms in terms of completely bounded polynomials, a counterexample to a direct analog of Theorem 1 was given for quartic polynomials in [3].

► **Theorem 2** (Arunachalam–Briët–Palazuelos). *For any $C > 0$, there exist an $n \in \mathbb{N}$ and a bounded quartic n -variable polynomial f such that no two-query quantum algorithm \mathcal{A} satisfies $\mathbb{E}[\mathcal{A}(x)] = Cf(x)$ for every $x \in \{-1, 1\}^n$.*

However, this result does not exclude the possibility that all bounded quartic polynomials can be (additively) approximated by two-query quantum algorithms. Moreover, the result is not constructive, relying on results from random matrix theory to show the existence of such polynomials. Finally, the result was obtained by transforming a certain random *cubic* polynomial into a quartic polynomial with similar properties. As we will explain here, the argument given in [3] to show that there is such a transformation contains an error. Here, we address these issues as follows:

First, we correct the error in [3], showing that Theorem 2 holds as stated.

Second, we give a completely explicit example for Theorem 2 using ideas from the field of additive combinatorics that were applied to construct counterexamples to certain far-reaching generalizations of the Grothendieck inequality [8].

Third, we strengthen Theorem 2 by showing that it still holds with a small additive error:

► **Theorem 3.** *For any $C > 0$, there exist an $n \in \mathbb{N}$, an $\varepsilon > 0$ and a bounded quartic n -variable polynomial f such that no two-query algorithm \mathcal{A} satisfies $|\mathbb{E}[\mathcal{A}(x)] - Cf(x)| < \varepsilon$ for every $x \in \{-1, 1\}^n$.*

This result is an application of a semidefinite-program (SDP) of Gribling and Laurent [11] for quantum query complexity. It can be interpreted as an analogue of results on approximate degree based on its linear-programming-based characterization (see for instance [9]). To the best of our knowledge, this is the first application of [11] to prove lower bounds on quantum query complexity. As such, we believe it can serve as a first step towards using this SDP to approach other problems such as proving large separations between approximate degree and quantum query complexity, for example [1].

In similar vein, we use a basic lower bound on the (real) Grothendieck constant, denoted K_G , based on the CHSH Bell inequality to give an impossibility result for one-query quantum algorithms. That is, we show that there exists a bounded quadratic polynomial f such that no one-query quantum algorithm approximates f with error less than $1 - 1/\sqrt{2}$.

Motivated by this, we pose as an open question whether this can be improved to $1 - 1/K_G$. Since the result of [2] achieves this for bounded bilinear forms, this would give yet another characterization of the Grothendieck constant. Tsirelson’s characterization in the context of Bell inequalities [18] being a famous example in quantum information theory, for instance.

We would like to remark that the characterization of quantum query algorithms given in [3], that we use here, was regarded as a nice, but unnatural result. However, it has gained relevance in recent times, as it has been proved to be an appropriate tool to make progress in a relevant question such as the Aaronson-Ambainis conjecture [5].

2 Preliminaries

Unless stated otherwise, below C will stand for an absolute positive constant whose value may change from line to line. All polynomials are assumed to be real and multivariate. A homogeneous polynomial is referred to as a *form*. A polynomial is multilinear if each variable appears with degree at most 1. Given an n -variate polynomial f and $p \in [1, \infty)$, define

$$\begin{aligned} \|f\|_p &= \left(\mathbb{E}_{x \in \{-1,1\}^n} f(x)^p \right)^{\frac{1}{p}} \\ \|f\|_\infty &= \max_{x \in \{-1,1\}^n} |f(x)|. \end{aligned}$$

We also define the following “commutative version” of a completely bounded norm:

$$\|f\|_{\text{iccb}} = \sup_{d \in \mathbb{N}} \left\{ \|f(A_1, \dots, A_n)\| : A_i \in \mathbb{C}^{d \times d}, \|A_i\| \leq 1, [A_i, A_j] = 0 \right\},$$

where the norms on the right-hand side are the usual operator norms.²

The following lemma [3, Theorem 1.3, Proposition 4.4] relates quantum query algorithms to completely bounded polynomials.

► **Lemma 4.** *Let \mathcal{A} be a t -query quantum algorithm. Then, there exists an $(n + 1)$ -variate form f of degree $2t$ such that $\|f\|_{\text{iccb}} \leq 1$ and which satisfies $f(x, 1) = \mathbb{E}[\mathcal{A}(x)]$ for every $x \in \{-1, 1\}^n$.*

We will also use a quantity associated specifically with multilinear cubic forms, that is polynomials of the form:

$$f(x) = \sum_{S \in \binom{[n]}{3}} c_S \prod_{i \in S} x_i, \tag{1}$$

where the c_S are some real coefficients. For $i \in [n]$, define the i th *slice* of f to be the symmetric matrix $M_i \in \mathbb{R}^{n \times n}$ with (j, k) -coefficient equal to $c_{\{i,j,k\}}$ if i, j, k are pairwise distinct and 0 otherwise. Then, define

$$\Delta(f) = \max_{i \in [n]} \|M_i\|.$$

The following is a slight variant of a decomposition due to Varopoulos [19].

² The notation iccb stands for “identical commutative completely bounded”, where the word identical distinguishes it from another natural variant of the completely bounded norm of a polynomial.

6:4 On Converses to the Polynomial Method

► **Lemma 5** (tri-linear Varopoulos decomposition). *Let f be an n -variate multilinear cubic form as in (1). Then, for some $d \in \mathbb{N}$, there exist pairwise commuting matrices $A_1, \dots, A_n \in \mathbb{R}^{d \times d}$ and orthogonal unit vectors $u, v \in \mathbb{R}^d$ such that $\|A_i\| \leq 1$, $[A_i, A_j] = 0$ and*

$$A_i^2 = 0 \tag{2}$$

$$\langle u, A_i v \rangle = 0 \tag{3}$$

$$\langle u, A_i A_j v \rangle = 0 \tag{4}$$

$$\langle u, A_i A_j A_k v \rangle = \frac{c_{\{i,j,k\}}}{\Delta(f)} \tag{5}$$

for all pairwise distinct $i, j, k \in [n]$.

Proof. For each $i \in [n]$, define M_i as above. Define $W_i = \Delta(f)^{-1} M_i$ and note that this has operator norm at most 1. For each $i \in [n]$, define the $(2n+2) \times (2n+2)$ block matrix

$$A_i = \begin{bmatrix} \text{---} & \text{---} & \text{---} & \text{---} \\ e_i & \text{---} & \text{---} & \text{---} \\ \text{---} & W_i^\top & \text{---} & \text{---} \\ \text{---} & \text{---} & e_i^\top & \text{---} \end{bmatrix},$$

where the first and last rows and columns have size 1, the second and third have size n and where the empty blocks are filled with zeros. Define $u = e_{2n+1}$ and $v = e_1$. The rest of the proof is identical to the proof of [8, Lemma 2.11], except for the property that $A_i^2 = 0$. This follows from the fact that

$$A_i^2 = \begin{bmatrix} \text{---} & \text{---} & \text{---} & \text{---} \\ \text{---} & \text{---} & \text{---} & \text{---} \\ W_i^\top e_i & \text{---} & \text{---} & \text{---} \\ \text{---} & e_i^\top W_i & \text{---} & \text{---} \end{bmatrix}.$$

and since the i th row and i th column of M_i are zero. ◀

► **Corollary 6.** *Let f be an n -variate multilinear cubic form as in (1). Suppose that an $(n+2)$ -variate quartic form $h \in \mathbb{R}[x_0, x_1, \dots, x_n, z]$ satisfies $h(x, 1) = x_0 f(x_1, \dots, x_n)$ for every $x \in \{-1, 1\}^{n+1}$. Then,*

$$\|h\|_{\text{iccb}} \geq \frac{\|f\|_2^2}{\Delta(f)}.$$

Proof. The multilinear monomials $\chi_S(x) = \prod_{i \in S} x_i$ with $S \subseteq \{0, \dots, n\}$ satisfy the orthogonality relations

$$\mathbb{E}_{x \in \{-1, 1\}^{n+1}} \chi_S(x) \chi_T(x) = \delta_{S,T}. \tag{6}$$

It follows that h and $x_0 f$ have equal coefficients for each quartic multilinear monomial in the variables x_0, \dots, x_n , which are c_S for $x_0 \chi_S$ with $S \in \binom{[n]}{3}$ and 0 otherwise. Let $A_1, \dots, A_n \in \mathbb{R}^{d \times d}$ and $u, v \in \mathbb{R}^d$ be as in Lemma 5 and let $A_0 = I, A_{n+1} = 0$. Commutativity and properties (2)–(4) imply that if a quartic monomial expression $A_i A_j A_k A_l$ with $i, j, k, l \in \{0, \dots, n+1\}$ has repeated indices or an index equal to $n+1$, then $\langle u, A_i A_j A_k A_l v \rangle = 0$.

With this, it follows from property (5) that

$$\begin{aligned}
 \langle u, h(A_0, \dots, A_{n+1})v \rangle &= \left\langle u, \sum_{S \in \binom{[n]}{3}} c_S A_0 \chi_S(A_1, \dots, A_n)v \right\rangle \\
 &= \sum_{S \in \binom{[n]}{3}} c_S \langle u, \chi_S(A_1, \dots, A_n)v \rangle \\
 &= \Delta(f)^{-1} \sum_{S \in \binom{[n]}{3}} c_S^2 \\
 &= \Delta(f)^{-1} \|f\|_2^2,
 \end{aligned} \tag{7}$$

where the last line is Parseval’s identity [13, Chapter 1]. ◀

3 Counterexamples

Here, we prove Theorems 2 and 3. But first we discuss the error in [3, pp. 920]. The proof there uses the equation

$$\sum_{\alpha, \beta \in \{0,1,2,3,4\}^n: |\alpha|+|\beta|=4} d'_{\alpha,\beta} x^\alpha = C \sum_{\alpha \in \{0,1\}^n: |\alpha|=4} d_\alpha x^\alpha \quad \forall x \in \{-1, 1\}^n, \tag{8}$$

where $d'_{\alpha,\beta}$, d_α and C are real numbers and $|\alpha|$ stands for $\sum_{i=1}^n \alpha_i$. It follows from (6) that $d'_{\alpha,0} = Cd_\alpha$ for all $\alpha \in \{0, 1\}^n$ such that $|\alpha| = 4$. What is used, however, is that $d'_{\alpha,0} = Cd_\alpha$ for all $\alpha \in \{0, 1, 2, 3, 4\}^n$ such that $|\alpha| = 4$, which is not true in general. For instance if $n = 2$, $C = 1$ and $d'_{(2,2),(0,0)} = d'_{(0,0),(4,0)} = -d'_{(2,0),(2,0)} = -d'_{(0,2),(2,0)} = 1$ and the rest of the coefficients set to 0, then (8) becomes $x_1^2 x_2^2 - x_1^4 - x_2^4 + 1 = 0$.

Corollary 6 gets around this issue by using a multilinear cubic form instead of just a cubic form. This results in matrices A_i in Lemma 5 that square to zero and has the effect that terms other than quartic multilinear monomials vanish in the left-hand side of (7).

3.1 A random example

The probabilistic proof of Theorem 2 uses a random cubic form as in (1) where the coefficients c_S are chosen to be independent uniformly distributed random signs. Parseval’s identity then gives $\|f\|_2^2 = \binom{n}{3}$. Each of the slices M_i of f is a random symmetric matrix with independent mean-zero entries of absolute value at most 1. A standard random-matrix inequality and the union bound then imply that $\Delta(f) \leq C\sqrt{n}$ with probability $1 - \exp(-Cn)$ [15, Corollary 2.3.6]. By Hoeffding’s inequality [7, Theorem 2.8] and the union bound, we have that $\|f\|_\infty \leq Cn^2$ with probability $1 - \exp(-Cn)$. Rescaling f then gives that there exists a bounded multilinear cubic form such that $\|f\|_2^2/\Delta(f) \geq C\sqrt{n}$. It now follows from Lemma 4 with Corollary 6 that the $(n + 1)$ -variable quartic polynomial $x_0 f(x_1, \dots, x_n)$ satisfies the requirements of Theorem 2.

3.2 An explicit example

We also give a constructive proof of Theorem 2 using techniques from [8], which were used there to disprove a conjecture of Pisier on certain far-reaching generalizations of the Grothendieck inequality. We do not exactly use the construction from that paper because it involves complex functions. Instead, we will use the Möbius function (defined below), which is real valued and has the desired properties.

6:6 On Converses to the Polynomial Method

Let n be a positive integer to be set later and let $f_0 : \mathbb{Z}_n \rightarrow [-1, 1]$ be a function to be set later (where as usual \mathbb{Z}_n denotes the group of integers modulo n). Define f to be the cubic multilinear form on $3n$ variables given by

$$f(x) = \sum_{a,b \in \mathbb{Z}_n} x(1, a)x(2, a + b)x(3, a + 2b)f_0(a + 3b), \quad (9)$$

where we indexed the variables by $[3] \times \mathbb{Z}_n$.

We claim that for some choice of f_0 , the quartic polynomial $x_0 f$, where x_0 is an additional variable, meets the requirements of Theorem 2. The generalized von Neumann inequality [17, Lemma 11.4] allows us to bound the ∞ -norm of f . For a function $g : \mathbb{Z}_n \rightarrow \mathbb{R}$ and $b \in \mathbb{Z}_n$, define its multiplicative derivative $\Delta_b g : \mathbb{Z}_n \rightarrow \mathbb{R}$ to be the function $\Delta_b g(a) = g(a + b)g(a)$. The Gowers 3-uniformity norm of g is then defined as

$$\|g\|_{U^3} = \left(\mathbb{E}_{a,b,c,d \in \mathbb{Z}_n} \Delta_b \Delta_c \Delta_d g(a) \right)^{\frac{1}{8}}.$$

► **Lemma 7** (generalized von Neumann inequality). *Suppose that n is coprime to 6. Then, for any function of the form (9), we have that*

$$\|f\|_{\infty} \leq n^2 \|f_0\|_{U^3}.$$

The polynomial f has $3n$ slices, $M_{i,a} \in \mathbb{R}^{[3] \times \mathbb{Z}_n}$ for each $i \in [3]$ and $a \in \mathbb{Z}_n$, which we view as 3×3 block-matrices with blocks indexed by \mathbb{Z}_n . The slice $M_{1,a}$ is supported only on the $(2, 3)$ and $(3, 2)$ blocks, which are each others' transposes. On its $(2, 3)$ block it has value $f_0(a + 3b)$ on coordinate $(a + b, a + 2b)$ for each b . In particular, this matrix has at most one nonzero entry in each row and column. It follows that a relabeling of the rows turns $M_{1,a}$ into a diagonal matrix with diagonal entries in $[-1, 1]$, and therefore $\|M_{1,a}\| \leq 1$. Similarly, we get that $\|M_{i,a}\| \leq 1$ for $i = 2, 3$. Hence, $\Delta(f) \leq 1$. Parseval's identity implies that

$$\|f\|_2^2 = n \sum_{a \in \mathbb{Z}_n} f_0(a)^2.$$

Identify \mathbb{Z}_n with $\{0, 1, \dots, n - 1\}$ in the standard way. We choose f_0 to be the Möbius function restricted to this interval. That is, set $f_0(0) = 0$ and for $a > 0$, set

$$f_0(a) = \begin{cases} 1 & \text{if } a \text{ is square-free with an even number of prime factors} \\ -1 & \text{if } a \text{ is square-free with an odd number of prime factors} \\ 0 & \text{otherwise.} \end{cases}$$

Tao and Teräväinen [16] recently proved that

$$\|f_0\|_{U^3} \leq \frac{1}{(\log \log n)^C}$$

for some absolute constant $C > 0$. It is also well-known that there are $\frac{6}{\pi^2}n - O(\sqrt{n})$ integers in $[n]$ that are square-free [12, page 269]. Normalizing f by $(\log \log n)^C/n^2$ and taking n coprime to 6 then gives a bounded multilinear cubic polynomial satisfying

$$\frac{\|f\|_2^2}{\Delta(f)} \geq \frac{6}{\pi^2} (\log \log n)^C - o(1).$$

This proves Theorem 2 as before.

► Remark 8. The *jointly completely bounded norm* of f is given by

$$\|f\|_{\text{jcb}} = \sup_{d \in \mathbb{N}} \|f(A_1, A_2, A_3)\|,$$

where the supremum is taken over maps $A_1, A_2, A_3 : \mathbb{Z}_n \rightarrow \mathbb{C}^{d \times d}$ such that $\|A_i(a)\| \leq 1$ and $[A_i(a), A_j(b)] = [A_i(a), A_j(b)^*] = 0$ for all $i \neq j$ and $a, b \in \mathbb{Z}_n$. Note that the only difference with the iccb norm defined in Section 2 is the second commutation relation involving the complex conjugates. This norm can also be stated in terms of tensor products and the supremum is attained by observable-valued maps. As such, this norm appears naturally in the context of non-local games. It was shown in [4] that Proposition 7 also holds for the jointly completely bounded norm, that is $\|f\|_{\text{jcb}} \leq n^2 \|f_0\|_{U^3}$. The proof of Corollary 6 easily implies that $\|f\|_{\text{iccb}} \geq \|f\|_2^2 / \Delta(f)$. This was used in [8] to prove that the jcb and iccb norms are inequivalent.

3.3 SDPs for quantum query complexity

Theorem 3 is based on an SDP for the completely bounded approximate degree of Gribling and Laurent [11]. The following notation will be convenient to state the SDP. Let $\mathcal{F}(n, t)$ be the set of functions $f : [n]^t \rightarrow \mathbb{R}$ of the form

$$f(\mathbf{i}) = \langle u, A_1(i_1) \cdots A_t(i_t)v \rangle,$$

where $u, v \in S^{d-1}$ and $A_1, \dots, A_t : [n] \rightarrow \{M \in \mathbb{R}^{d \times d} : \|M\| \leq 1\}$ for some $d \in \mathbb{N}$. A basic linear algebra argument shows that any such function can be obtained by setting $d = n^t$. Given a function $\phi : \{-1, 1\}^n \rightarrow \mathbb{R}$, a sequence $\mathbf{i} \in [n+1]^t$ and setting $x_{n+1} = 1$, define

$$\hat{\phi}(\mathbf{i}) = \mathbb{E}_{x \in \{-1, 1\}^n} \phi(x) \prod_{j=1}^t x_{i_j}.$$

Note that if

$$\phi(x) = \sum_{S \in \binom{[n]}{t}} c_S \chi_S(x)$$

is a multilinear form of degree t , then

$$\hat{\phi}(\mathbf{i}) = \begin{cases} c_S & \text{if } \{i_1, \dots, i_t\} = S \\ 0 & \text{otherwise.} \end{cases} \quad (10)$$

Given $f : \{-1, 1\}^n \rightarrow [-1, 1]$ and $t \in \mathbb{N}$, define

$$\begin{aligned} \text{SDP}(f, t) = \max \quad & \mathbb{E}_{x \in \{-1, 1\}^n} \phi(x) f(x) - w \\ \text{s.t.} \quad & \phi : \{-1, 1\}^n \rightarrow \mathbb{R}, w \in \mathbb{R} \\ & \|\phi\|_1 = 1 \\ & (1/w)\hat{\phi} \in \mathcal{F}(n+1, t). \end{aligned} \quad (11)$$

Program (11) corresponds to the optimization problem (24) of [11] for total functions and is written in a more convenient way for our purposes. There, f is considered to take values in $\{-1, 1\}$, but their results still hold if f is allowed to take values in \mathbb{R} , as we do here. Also, we must point out that the $A_i(i_s)$ used in the program (24) of [11] are unitaries, but there is no problem if we substitute them by contractions, thanks to the fact that every contraction can be seen as the top left corner of an unitary matrix [2, Lemma 7].

► **Theorem 9** (Gribling-Laurent). *If the optimal value of program (11) is strictly larger than ε , then there is no $\lceil t/2 \rceil$ -query algorithm \mathcal{A} such that $|\mathbb{E}(\mathcal{A}(x)) - f(x)| \leq \varepsilon$.*

3.4 Approximation of quadratic forms

Theorem 1 implies that bounded quadratic polynomials can be approximated by one-query quantum algorithms with error at most $1 - C$. Moreover, for $2n$ -variate bounded bilinear forms $f(x, y) = x^\top Ay$ for $A \in \mathbb{R}^{n \times n}$, we can C to be $1/K_G(n)$, where $K_G(n)$ is the real Grothendieck constant of dimension n (see [3] for a short proof). Then, bounded bilinear forms can be approximated with an additive error of at most $1 - 1/K_G(n)$. Using Theorem 9, we show that this is optimal for $n = 2$, in which case $K_G(2) = \sqrt{2}$ [10].

► **Proposition 10.** *There exists a bilinear form $f \in \mathbb{R}[x_1, x_2, x_3, x_4]$ such that there is no one-query quantum algorithm that approximates f on every $x \in \{-1, 1\}^4$ with an additive error smaller than $1 - 1/\sqrt{2}$.*

Proof. We use the bilinear form that attains the Grothendieck constant of dimension 2, which is captured by the CHSH game. This form $f \in \mathbb{R}[x_1, x_2, x_3, x_4]$ is given by

$$f(x) = \frac{1}{2}(x_1(x_3 + x_4) + x_2(x_3 - x_4)).$$

Clearly f maps $\{-1, 1\}^4$ to $\{-1, 1\}$, and so $\|f\|_1 = \|f\|_2^2 = 1$. We now emulate the construction from Lemma 5. Writing the coefficients of f as c_S for $S \in \binom{[4]}{2}$, for each $i \in [4]$ define the unit vector $w_i \in \mathbb{R}^4$ by

$$w_i = \frac{1}{\sqrt{2}} \sum_{j \in [4] \setminus \{i\}} c_{\{i,j\}} e_j.$$

Now define the matrices $A_i \in \mathbb{R}^{6 \times 6}$ by

$$A(i) = \begin{pmatrix} 0 & 0 & 0 \\ w_i & 0 & 0 \\ 0 & e_i^\top & 0 \end{pmatrix}.$$

It is easily verified that $A(i)^2 = 0$ and that the $(6, 1)$ -coordinate of $A(i)A(j)$ equals $c_{\{i,j\}}/\sqrt{2}$ if $i \neq j$, from which it also follows that these matrices commute. Setting $A(5) = 0$, we get that

$$\langle e_6, A(i)A(j)e_1 \rangle = \begin{cases} \frac{c_{\{i,j\}}}{\sqrt{2}} & \text{if } \{i, j\} \in \binom{[4]}{2} \\ 0 & \text{otherwise.} \end{cases}$$

Setting $\phi = f$ then gives that $\sqrt{2}\phi \in \mathcal{F}(5, 2)$ and $\|\phi\|_1 = 1$. This shows that $\text{SDP}(f, 2) \geq 1 - 1/\sqrt{2}$. ◀

Proposition 10 leads to a following natural question:

► **Question 1.** *Is it true that for any $\epsilon > 0$ there are an integer n and a bounded bilinear form $f \in \mathbb{R}[x_1, \dots, x_{2n}]$ such that there is no one-query quantum algorithm that approximates f on every $x \in \{-1, 1\}^{2n}$ with an error smaller than $1 - \frac{1}{K_G} - \epsilon$?*

3.5 Approximation of cubic forms

Given that a generalization of Theorem 1 has been ruled out for quartic polynomials, one may wonder if a weaker converse for the polynomial method is possible:

► **Question 2.** Are there constants $C > 0$ and $\varepsilon > 0$ such that for every bounded polynomial f of degree 4 there is a 2-query algorithm \mathcal{A} such that $|\mathbb{E}(\mathcal{A}(x)) - Cf(x)| < \varepsilon$ for every $x \in \{-1, 1\}^n$?

An affirmative answer to this question would imply that every polynomial of degree 4 could be approximated by a 2-query algorithm with additive error $1 - C + \varepsilon$. This would be the converse for the polynomial method that motivated Theorem 1 in [2]. Theorem 3 means that the ε appearing in Question 2 cannot be arbitrarily small. In other words, Theorem 3 says that there is no multiplicative converse even if we allow an (arbitrarily) small additive error.

Proof of Theorem 3. Let $f \in \mathbb{R}[x_1, \dots, x_n]$ be a bounded multilinear cubic form as in (1). As shown in the proof of Corollary 6, there exist unit vectors $u, v \in \mathbb{R}^d$ and mappings $A : \{0, 1, \dots, n+1\} \rightarrow \mathbb{R}^{d \times d}$ such that $\|A(i)\| \leq 1$ for each i and

$$\langle u, A(i)A(j)A(k)A(l)v \rangle = \begin{cases} \frac{cs}{\Delta(f)} & \text{if } \{i, j, k, l\} = \{0\} \cup S \text{ for } S \in \binom{[n]}{3} \\ 0 & \text{otherwise.} \end{cases}$$

Let $g = x_0 f / \|f\|_\infty$. Then, the function $\phi = x_0 f / \|f\|_1$ meets the criteria of (11) with $w = \Delta(f) / \|f\|_1$ and shows that

$$\begin{aligned} \text{SDP}(g, 4) &\geq \frac{\|f\|_2^2}{\|f\|_1 \|f\|_\infty} - \frac{\Delta(f)}{\|f\|_1} \\ &\geq \frac{\|f\|_2^2}{\|f\|_1 \|f\|_\infty} \left(1 - \frac{\Delta(f) \|f\|_\infty}{\|f\|_2^2}\right). \end{aligned}$$

If f is the random example from Section 3.1, then $\|f\|_2^2 = \binom{n}{3}$ and $\Delta(f) \|f\|_\infty \leq Cn^{5/2}$ with high probability. In particular, the above is positive for sufficiently large n . Similarly, for any $C \in (0, 1)$ we get that $\text{SDP}(Cg, 4) > 0$ for sufficiently large n . The result now follows from Theorem 9. ◀

References

- 1 Scott Aaronson. Open problems related to quantum query complexity. *ACM Transactions on Quantum Computing*, 2(4):1–9, 2021.
- 2 Scott Aaronson, Andris Ambainis, Jānis Iraids, Martins Kokainis, and Juris Smotrovs. Polynomials, quantum query complexity, and Grothendieck’s inequality. In *31st Conference on Computational Complexity, CCC 2016*, pages 25:1–25:19, 2016. [arXiv:1511.08682](#).
- 3 Srinivasan Arunachalam, Jop Briët, and Carlos Palazuelos. Quantum query algorithms are completely bounded forms. *SIAM J. Comput.*, 48(3):903–925, 2019. Preliminary version in ITCS’18.
- 4 Tom Bannink, Jop Briët, Harry Buhrman, Farrokh Labib, and Troy Lee. Bounding Quantum-Classical Separations for Classes of Nonlocal Games. In *36th International Symposium on Theoretical Aspects of Computer Science (STACS 2019)*, volume 126 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 12:1–12:11, Dagstuhl, Germany, 2019. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. Available at [arXiv:1811.11068](#). doi:10.4230/LIPIcs.STACS.2019.12.
- 5 Nikhil Bansal, Makrand Sinha, and Ronald de Wolf. Influence in completely bounded block-multilinear forms and classical simulation of quantum algorithms. *arXiv preprint*, 2022. [arXiv:2203.00212](#).
- 6 Robert Beals, Harry Buhrman, Richard Cleve, Michele Mosca, and Ronald de Wolf. Quantum lower bounds by polynomials. *J. ACM*, 48(4):778–797, 2001. doi:10.1145/502090.502097.


6:10 On Converses to the Polynomial Method

- 7 S. Boucheron, G. Lugosi, and P. Massart. *Concentration inequalities: A nonasymptotic theory of independence*. Oxford university press, 2013.
- 8 Jop Briët and Carlos Palazuelos. Failure of the trilinear operator space Grothendieck inequality. *Discrete Analysis*, 2019. Paper No. 8.
- 9 Mark Bun, Robin Kothari, and Justin Thaler. The polynomial method strikes back: Tight quantum query bounds via dual polynomials. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing (STOC'18)*, pages 297–310, 2018.
- 10 Peter C Fishburn and James A Reeds. Bell inequalities, Grothendieck’s constant, and root two. *SIAM Journal on Discrete Mathematics*, 7(1):48–56, 1994.
- 11 Sander Gribling and Monique Laurent. Semidefinite programming formulations for the completely bounded norm of a tensor. *arXiv preprint*, 2019. [arXiv:1901.04921](https://arxiv.org/abs/1901.04921).
- 12 Godfrey Harold Hardy, Edward Maitland Wright, et al. *An introduction to the theory of numbers*. Oxford university press, 1979.
- 13 Ryan O’Donnell. *Analysis of Boolean Functions*. Cambridge University Press, 2009. doi:10.1017/cbo9781139814782.
- 14 Gilles Pisier. Grothendieck’s theorem, past and present. *Bulletin of the American Mathematical Society*, 49(2):237–323, 2012.
- 15 T. Tao. *Topics in Random Matrix Theory*. Graduate studies in mathematics. American Mathematical Society, 2012. URL: <https://books.google.nl/books?id=L51VAwAAQBAJ>.
- 16 Terence Tao and Joni Teräväinen. Quantitative bounds for Gowers uniformity of the Möbius and von Mangoldt functions. *arXiv preprint*, 2021. [arXiv:2107.02158](https://arxiv.org/abs/2107.02158).
- 17 Terence Tao and Van H Vu. *Additive combinatorics*, volume 105. Cambridge University Press, 2006.
- 18 B. S. Tsirelson. Quantum generalizations of Bell’s inequality. *Letters in Mathematical Physics*, 1980. doi:10.1007/BF00417500.
- 19 N. Th. Varopoulos. On an inequality of von Neumann and an application of the metric theory of tensor products to operators theory. *J. Functional Analysis*, 16:83–100, 1974. doi:10.1016/0022-1236(74)90071-8.

The Quantum Approximate Optimization Algorithm at High Depth for MaxCut on Large-Girth Regular Graphs and the Sherrington-Kirkpatrick Model

Joao Basso   

Google Quantum AI, Venice, CA, USA

Edward Farhi 

Google Quantum AI, Venice, CA, USA

Center for Theoretical Physics, Massachusetts Institute of Technology, Cambridge, MA, USA

Kunal Marwaha   

Department of Computer Science, University of Chicago, IL, USA

Benjamin Villalonga  

Google Quantum AI, Venice, CA

Leo Zhou  

Walter Burke Institute for Theoretical Physics, California Institute of Technology, Pasadena, CA, USA

Abstract

The Quantum Approximate Optimization Algorithm (QAOA) finds approximate solutions to combinatorial optimization problems. Its performance monotonically improves with its depth p . We apply the QAOA to MaxCut on large-girth D -regular graphs. We give an iterative formula to evaluate performance for any D at any depth p . Looking at random D -regular graphs, at optimal parameters and as D goes to infinity, we find that the $p = 11$ QAOA beats all classical algorithms (known to the authors) that are free of unproven conjectures. While the iterative formula for these D -regular graphs is derived by looking at a single tree subgraph, we prove that it also gives the ensemble-averaged performance of the QAOA on the Sherrington-Kirkpatrick (SK) model defined on the complete graph. We also generalize our formula to Max- q -XORSAT on large-girth regular hypergraphs. Our iteration is a compact procedure, but its computational complexity grows as $O(p^2 4^p)$. This iteration is more efficient than the previous procedure for analyzing QAOA performance on the SK model, and we are able to numerically go to $p = 20$. Encouraged by our findings, we make the optimistic conjecture that the QAOA, as p goes to infinity, will achieve the Parisi value. We analyze the performance of the quantum algorithm, but one needs to run it on a quantum computer to produce a string with the guaranteed performance.

2012 ACM Subject Classification Theory of computation \rightarrow Quantum computation theory; Theory of computation \rightarrow Approximation algorithms analysis; Mathematics of computing \rightarrow Combinatorial optimization

Keywords and phrases Quantum algorithm, Max-Cut, spin glass, approximation algorithm

Digital Object Identifier 10.4230/LIPIcs.TQC.2022.7

Related Version *Full Version:* <https://arxiv.org/abs/2110.14206> [4]

Funding *Kunal Marwaha:* National Science Foundation Graduate Research Fellowship Program under Grant No. DGE-1746045. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.



© Joao Basso, Edward Farhi, Kunal Marwaha, Benjamin Villalonga, and Leo Zhou; licensed under Creative Commons License CC-BY 4.0

17th Conference on the Theory of Quantum Computation, Communication and Cryptography (TQC 2022).

Editors: François Le Gall and Tomoyuki Morimae; Article No. 7; pp. 7:1–7:21

Leibniz International Proceedings in Informatics



LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Acknowledgements The authors thank Sam Gutmann for being there and Matthew P. Harrigan for a careful read of the manuscript.

1 Introduction

We are at the start of an era in which quantum devices are running algorithms. We need to understand the power of quantum computers for solving or finding approximate solutions to combinatorial optimization problems. One approach is to learn by experimenting on hardware. Although useful for probing the hardware and testing algorithms at small sizes, it does not give a convincing picture of asymptotic behavior. To this end we need mathematical studies of the behavior of quantum algorithms, running on ideal circuits, at large sizes. In this paper we take a step in that direction by analyzing the Quantum Approximate Optimization Algorithm as applied to a certain combinatorial optimization problem. The instances are large and the depth of the algorithm is high. For this task, we will see that the QAOA outperforms the best assumption-free classical algorithm.

MaxCut is a combinatorial optimization problem on bit strings whose input is a graph. Each bit is associated with a vertex, and the goal is to maximize the number of edges with bit assignments that disagree on the two ends of the edge. It is NP-hard to solve this problem exactly, and even approximating the optimal solution beyond a certain ratio is NP-hard [30]. We focus on MaxCut for large-girth D -regular graphs. On these graphs, the currently known best classical algorithms (including Goemans-Williamson and the Gaussian wave process [23, 20, 3, 29]) achieve an average-case cut fraction (the number of cut edges output by the algorithm divided by the number of edges) of $1/2 + (2/\pi)/\sqrt{D}$ as both the girth and D go to ∞ , where $2/\pi \approx 0.6366$.

We apply the Quantum Approximate Optimization Algorithm (QAOA) [15] to large-girth D -regular graphs. The QAOA depends on a parameter p , the algorithm’s depth. At small p , the QAOA has been realized in current quantum hardware [19]. Some analytic results are also known. At $p = 1$, the QAOA has a guaranteed approximation ratio (the number of cut edges output by the algorithm divided by the maximum number of edges that can be cut) of at least 0.6924 on all 3-regular graphs [15] and an expected cut fraction of at least $1/2 + 0.3032/\sqrt{D}$ on triangle-free graphs [31]. For $p = 2$, the QAOA has an approximation ratio of at least 0.7559 on 3-regular graphs with girth more than 5 and, for $p = 3$, that ratio becomes 0.7924 when the girth is more than 7 [32]. So far, expressions for the QAOA’s performance on any fixed- D regular, large-girth graph are known only for $p = 1$ [31] and $p = 2$ [21].

In this work, we analyze the performance of the QAOA on any large-girth D -regular graph for any choice of p by looking at a single tree subgraph. Using the regularity of this tree subgraph, we derive an iteration that computes the performance of the QAOA. After optimizing over the $2p$ input parameters, we find that the $p = 11$ QAOA improves on $1/2 + (2/\pi)/\sqrt{D}$, when D is large and the girth is more than 23. This is better than all assumption-free classical algorithms known to the authors.¹

We also show that this performance, obtained from one subgraph, is mathematically equal to the ensemble-averaged performance of the QAOA applied to the Sherrington-Kirkpatrick

¹ There is a recent classical message-passing algorithm [1] that also does better than $1/2 + (2/\pi)/\sqrt{D}$ for MaxCut on large-girth D -regular graphs. It gets asymptotically close to the optimum assuming the solution space has no “overlap gap property” (see [17] for a review).

(SK) model [16]. This implies that the iteration in this paper can also be used to give the QAOA's performance on the SK model. A recent related work can be found in Ref. [8]. Our iteration is more efficient than the one originally shown in Ref. [16], and we have been able to go numerically to higher depth.

Encouraged by our findings, we conjecture that the large p performance of the QAOA will achieve the optimal cut fraction on large random D -regular graphs, where a vanishing fraction of neighborhoods are not locally tree-like. The optimal cut fraction on these graphs is also related to the SK model. It is $1/2 + \Pi_*/\sqrt{D} + o(1/\sqrt{D})$, where $\Pi_* = 0.763166\dots$, the Parisi value, is the ground state energy density of the SK model [25, 12]. If our conjecture is right we have a simple, though computationally intensive, new iteration for calculating Π_* .

Generalizing our formalism, we also analyze the performance of the QAOA for Max- q -XORSAT (of which MaxCut is a special case at $q = 2$) on large-girth D -regular hypergraphs. The $p = 1$ QAOA was recently found to do better than an analogous classical threshold algorithm for $q > 4$ [22]. The iterative formula for general q is very similar to that for MaxCut and has the same time and memory complexities in the $D \rightarrow \infty$ limit. We run this iteration to find optimal QAOA parameters and performance for $3 \leq q \leq 6$ and $1 \leq p \leq 14$. Moreover, we discuss potential obstructions to the QAOA from not "seeing" the whole graph.

The paper is organized as follows. In Section 2, we introduce the necessary definitions to describe the QAOA and the MaxCut problem. In Section 3, we describe two iterations that compute the performance of the QAOA for MaxCut on large-girth D -regular graphs at fixed depth: one for finite D and the other for $D \rightarrow \infty$ (proof in Appendix A). We also present our results from numerical evaluation and optimization of the QAOA objective function up to $p = 20$. In Section 4, we argue that the performance of the QAOA on large-girth regular graphs and on the SK model are equivalent. We conjecture in Section 5 that the iteration in Section 3.2 for infinite D is an alternative procedure to compute the Parisi value. In Section 6, we generalize our iterations to evaluate the QAOA's performance for Max- q -XORSAT on large-girth regular hypergraphs. Finally, in Section 7 we discuss our results and suggest some future avenues of work.

2 Background on the QAOA and MaxCut

The QAOA [15] is a quantum algorithm for finding approximate solutions to combinatorial optimization problems. The cost function counts the number of clauses satisfied by an input string. Given a cost function $C(\mathbf{z})$ on strings $\mathbf{z} \in \{\pm 1\}^n$, we can define a corresponding quantum operator, diagonal in the computational basis, as $C|\mathbf{z}\rangle = C(\mathbf{z})|\mathbf{z}\rangle$. Moreover, let $B = \sum_{j=1}^n X_j$, where X_j is the Pauli X operator acting on qubit j . Let $\boldsymbol{\gamma} = (\gamma_1, \gamma_2, \dots, \gamma_p)$ and $\boldsymbol{\beta} = (\beta_1, \beta_2, \dots, \beta_p)$. The QAOA initializes the system of qubits in the state $|s\rangle = |+\rangle^{\otimes n}$ and applies p alternating layers of $e^{-i\gamma_j C}$ and $e^{-i\beta_j B}$ to prepare the state

$$|\boldsymbol{\gamma}, \boldsymbol{\beta}\rangle = e^{-i\beta_p B} e^{-i\gamma_p C} \dots e^{-i\beta_1 B} e^{-i\gamma_1 C} |s\rangle. \quad (2.1)$$

For a given cost function C , the corresponding QAOA objective function is $\langle \boldsymbol{\gamma}, \boldsymbol{\beta} | C | \boldsymbol{\gamma}, \boldsymbol{\beta} \rangle$. Preparing the quantum state $|\boldsymbol{\gamma}, \boldsymbol{\beta}\rangle$ and then measuring in the computational basis enough times, one will find a bit string \mathbf{z} such that $C(\mathbf{z})$ is near $\langle \boldsymbol{\gamma}, \boldsymbol{\beta} | C | \boldsymbol{\gamma}, \boldsymbol{\beta} \rangle$ or better.

We study the performance of the QAOA on MaxCut. Given a graph $G = (V, E)$ with vertices in V and edges in E , the MaxCut cost function is

$$C_{\text{MC}}(\mathbf{z}) = \sum_{(u,v) \in E} \frac{1}{2} (1 - z_u z_v). \quad (2.2)$$

7:4 QAOA at High Depth for MaxCut on Large-Girth Regular Graphs & SK Model

We restrict our attention to graphs that are regular and have girth greater than $2p + 1$. We work with these graphs because the subgraph that the QAOA at depth p sees on them are regular trees and this enables our calculation. Here, by “seeing” we refer to the fact that the output of the QAOA on a qubit depends only on a neighborhood of qubits that are within distance p to the given qubit on the graph. In what follows, we focus on $(D + 1)$ -regular graphs, which implies the subgraph seen by the QAOA on each edge is a D -ary tree.

With D large, we will see that the optimal γ are of order $1/\sqrt{D}$. So we find it convenient to prepare the QAOA state $|\gamma, \beta\rangle$ using the scaled cost function operator

$$C = -\frac{1}{\sqrt{D}} \sum_{(u,v) \in E} Z_u Z_v, \quad (2.3)$$

where we have subtracted a constant that only introduces an irrelevant phase. The factor of $1/2$ has been dropped so that this form of the cost function will match the cost function used in the Sherrington-Kirkpatrick model. Note we are preparing the state $|\gamma, \beta\rangle$ using C as a driver instead of the C_{MC} operator. With this scaling, the optimal γ will be of order unity instead of $1/\sqrt{D}$.

Given any edge in a $(D + 1)$ -regular graph with girth greater than $2p + 1$ the subgraph with vertices at most p away from the edge is a D -ary tree regardless of which edge. Since the QAOA at depth p only sees these trees, we have

$$\langle \gamma, \beta | C_{\text{MC}} | \gamma, \beta \rangle = \frac{1}{2} |E| \left(1 - \langle \gamma, \beta | Z_u Z_v | \gamma, \beta \rangle \right) \quad (2.4)$$

where $(u, v) \in E$ is any edge. The cut fraction output by the QAOA is then

$$\frac{\langle \gamma, \beta | C_{\text{MC}} | \gamma, \beta \rangle}{|E|} = \frac{1}{2} - \frac{1}{2} \langle \gamma, \beta | Z_u Z_v | \gamma, \beta \rangle. \quad (2.5)$$

Since the QAOA cannot beat the optimal cut fraction of $1/2 + \text{order}(1/\sqrt{D})$ in a typical random regular graph, we write

$$\frac{\langle \gamma, \beta | Z_u Z_v | \gamma, \beta \rangle}{2} = -\frac{\nu_p(D, \gamma, \beta)}{\sqrt{D}} \quad (2.6)$$

where $\nu_p(D, \gamma, \beta)$ for good parameters will be of order unity.

3 The QAOA on large-girth $(D + 1)$ -regular graphs

We describe two iterations to evaluate the performance of the QAOA at high depth on MaxCut on large-girth $(D + 1)$ -regular graphs. The cut fraction output by the QAOA at any parameters is

$$\frac{\langle \gamma, \beta | C_{\text{MC}} | \gamma, \beta \rangle}{|E|} = \frac{1}{2} + \frac{\nu_p(D, \gamma, \beta)}{\sqrt{D}}. \quad (3.1)$$

We give one iteration to evaluate $\nu_p(D, \gamma, \beta)$ at finite D , and one for the $D \rightarrow \infty$ limit. We have attempted to make this section self-contained for those readers only interested in the form of the iterations, and deferred the detailed proofs of these iterations to Appendix A.

In what follows, we index vectors in the following order:

$$\mathbf{a} = (a_1, a_2, \dots, a_p, a_0, a_{-p}, \dots, a_{-2}, a_{-1}). \quad (3.2)$$

Define, for $1 \leq r \leq p$,

$$\Gamma_r = \gamma_r, \quad \Gamma_0 = 0, \quad \Gamma_{-r} = -\gamma_r. \quad (3.3)$$

That is, Γ is a $(2p+1)$ -component vector. Furthermore, let

$$\begin{aligned} f(\mathbf{a}) = & \frac{1}{2} \langle a_1 | e^{i\beta_1 X} | a_2 \rangle \cdots \langle a_{p-1} | e^{i\beta_{p-1} X} | a_p \rangle \langle a_p | e^{i\beta_p X} | a_0 \rangle \\ & \times \langle a_0 | e^{-i\beta_p X} | a_{-p} \rangle \langle a_{-p} | e^{-i\beta_{p-1} X} | a_{-(p-1)} \rangle \cdots \langle a_{-2} | e^{-i\beta_1 X} | a_{-1} \rangle \end{aligned} \quad (3.4)$$

where $a_i \in \{+1, -1\}$ enumerates the two computational basis states, and

$$\langle a_1 | e^{i\beta X} | a_2 \rangle = \begin{cases} \cos(\beta) & \text{if } a_1 = a_2 \\ i \sin(\beta) & \text{if } a_1 \neq a_2. \end{cases} \quad (3.5)$$

3.1 An iteration for any finite D

Here we give an iteration that allows us to evaluate $\nu_p(D, \gamma, \beta)$ for any input parameters and D .

Let $H_D^{(m)}: \{-1, 1\}^{2p+1} \rightarrow \mathbb{C}$ for $0 \leq m \leq p$. We start with $H_D^{(0)}(\mathbf{a}) = 1$ and let

$$H_D^{(m)}(\mathbf{a}) = \left(\sum_{\mathbf{b}} f(\mathbf{b}) H_D^{(m-1)}(\mathbf{b}) \cos \left[\frac{1}{\sqrt{D}} \Gamma \cdot (\mathbf{a}\mathbf{b}) \right] \right)^D \quad \text{for } 1 \leq m \leq p \quad (3.6)$$

where we denote $\mathbf{a}\mathbf{b}$ as the entry-wise product, i.e. $(\mathbf{a}\mathbf{b})_j = a_j b_j$. By starting with $H_D^{(0)}(\mathbf{a}) = 1$ and iteratively evaluating Eq. (3.6) for $m = 1, 2, \dots, p$, we arrive at $H_D^{(p)}(\mathbf{a})$ that can be used to compute

$$\nu_p(D, \gamma, \beta) = \frac{i\sqrt{D}}{2} \sum_{\mathbf{a}, \mathbf{b}} a_0 b_0 f(\mathbf{a}) f(\mathbf{b}) H_D^{(p)}(\mathbf{a}) H_D^{(p)}(\mathbf{b}) \sin \left[\frac{1}{\sqrt{D}} \Gamma \cdot (\mathbf{a}\mathbf{b}) \right]. \quad (3.7)$$

We prove this in Appendix A.1. The key idea is to use the fact that when girth $> 2p+1$, the subgraph seen by the QAOA is a pair of D -ary trees of p levels glued at their roots (see Figure 1(a) for an example). Then ν_p is given as a sum over all $O(D^p)$ nodes in this subgraph. Since every node in the tree has exactly D children that couples to their parent in exactly the same way, we can greatly simplify the process by summing from the leaves of the tree, then their parents, and their parents' parents, and so on. This yields a p -step iteration where at each step $m = 1, 2, \dots, p$, we have a compact description of the contributions of the nodes from the bottom $m-1$ levels via $H_D^{(m-1)}$ (see Figure 1(b)).

Note that each step of the above iteration involves a sum with 2^{2p+1} terms for each of the 2^{2p+1} entries of $H_D^{(m)}(\mathbf{a})$. The final step has a sum with $O(16^p)$ terms. Overall, this iteration has a time complexity of $O(p 16^p)$ and a memory complexity of $O(4^p)$. This is much faster than the original ‘‘light cone’’ approach that directly evaluates $\langle Z_u Z_v \rangle$ on the subgraph seen by the QAOA [15]. That procedure takes $2^{O(D^p)}$ time without utilizing the symmetric structure of the regular tree subgraph.

3.2 An iteration for $D \rightarrow \infty$

We find that in the infinite D limit we get a more compact iteration which takes fewer steps to evaluate. We state the result here and prove it in Appendix A.2.

Define matrices $G^{(m)} \in \mathbb{C}^{(2p+1) \times (2p+1)}$ for $0 \leq m \leq p$ as follows. For $j, k \in \{1, \dots, p, 0, -p, \dots, -1\}$, let $G_{j,k}^{(0)} = \sum_{\mathbf{a}} f(\mathbf{a}) a_j a_k$, and

$$G_{j,k}^{(m)} = \sum_{\mathbf{a}} f(\mathbf{a}) a_j a_k \exp\left(-\frac{1}{2} \sum_{j', k' = -p}^p G_{j', k'}^{(m-1)} \Gamma_{j'} \Gamma_{k'} a_{j'} a_{k'}\right) \quad \text{for } 1 \leq m \leq p. \quad (3.8)$$

Starting at $m = 0$ and going up by p steps, we arrive at $G^{(p)}$ which is used to compute

$$\nu_p(\gamma, \beta) := \lim_{D \rightarrow \infty} \nu_p(D, \gamma, \beta) = \frac{i}{2} \sum_{j=-p}^p \Gamma_j (G_{0,j}^{(p)})^2. \quad (3.9)$$

Since there are $p + 1$ matrices with $O(p^2)$ entries, and each involves a sum over $O(4^p)$ terms, this iteration naïvely has a time complexity of $O(p^3 4^p)$. This is quadratically better than the time complexity of the finite- D formula. The memory complexity is only $O(p^2)$ for storing the $G^{(m)}$ matrix, which is exponentially better than $O(4^p)$ memory needed to store the entries of $H_D^{(m)}$ in the finite- D iteration.

We note some properties about this iteration. Superficially Eq. (3.8) looks like a recursive map on the matrices $G^{(m)}$ which one might think would only asymptotically converge in the number of steps. However it converges to a fixed point $G^{(p)}$ after p steps in a highly structured way. In particular, the iteration has the following three sets of properties, whose proof can be found in Ref. [4, Appendix A]. We use the convention $1 \leq r < s \leq p$ and $j, k \in \{1, \dots, p, 0, -p, \dots, -1\}$.

(a) Values of the diagonal and anti-diagonal of $G^{(m)}$ are all 1. $G^{(m)}$ is symmetric with respect to the diagonal, reflection with respect to the anti-diagonal results in complex conjugation, and the matrix consists of 8 triangular regions which are rotations, reflections, and/or complex conjugations of each other. To be precise, $G^{(m)}$ satisfies the following properties:

$$\begin{aligned} (1) \quad G_{j,k}^{(m)} &= G_{k,j}^{(m)} & (3) \quad G_{0,r}^{(m)} &= G_{0,-r}^{(m)*} \\ (2) \quad G_{j,j}^{(m)} &= G_{j,-j}^{(m)} = 1 & (4) \quad G_{r,s}^{(m)} &= G_{r,-s}^{(m)} = G_{-r,-s}^{(m)*} = G_{-r,s}^{(m)*} \end{aligned}$$

These are sketched in Figure 1(c).

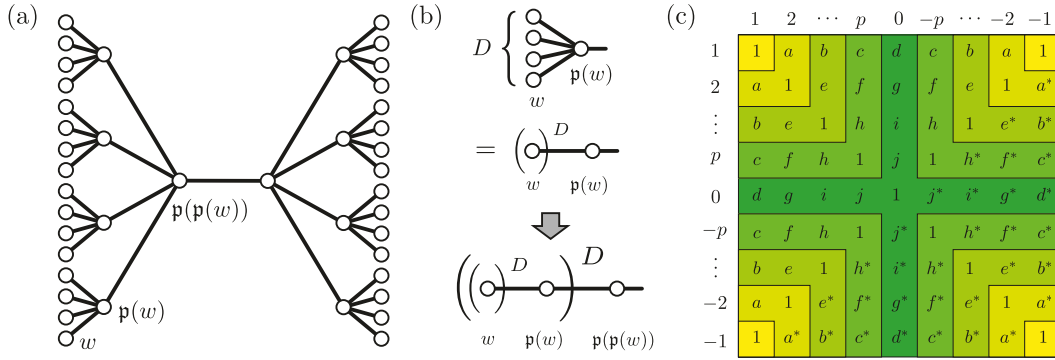
- (b) $G_{r,s}^{(m)}$ only depends on $G_{r',s'}^{(m-1)}$ where $1 \leq r' < s' < s$. Similarly, $G_{0,r}^{(m)}$ only depends on $G_{r',s'}^{(m-1)}$ for $1 \leq r' < s' \leq p$.
- (c) As a consequence of (b), at each step m of the iteration the corner blocks of size $(m+1) \times (m+1)$ of $G^{(m)}$ converge to their final value, i.e., they reach a fixed point and do not change in later iteration steps. This implies that matrix $G^{(p)}$ is a fixed point. This is sketched in Figure 1(c), where matrix entries of the same color reach their fixed point at the same step of the iteration, starting from the corners and ending with the central “cross” at step p .

Making use of (b) and some properties of $f(\mathbf{a})$ allows us to lower the complexity of the iterative procedure to $O(p^2 4^p)$. For more details, see Ref. [4, Appendix A.4].

3.3 Numerical evaluation and optimization for the $D \rightarrow \infty$ limit

Let

$$\bar{\nu}_p = \max_{\gamma, \beta} \nu_p(\gamma, \beta). \quad (3.10)$$



■ **Figure 1** (a) Example tree subgraph seen by the QAOA at $p = 2$ on a large-girth regular graph. For any node w on the tree, we denote $p(w)$ as its parent. (b) A visualization of our iteration for finite D . (c) Sketch of the properties of matrices $G^{(m)}$ in our iteration for $D \rightarrow \infty$, at $p = 4$. Regions of the same color converge in the same iteration step, starting from the corners and with the central row and column converging after p steps.

Numerically implementing the iteration summarized in Section 3.2 and optimizing for γ, β we find $\bar{\nu}_p$ up to $p = 17$. The values are given in Table 1 and plotted in Figure 2 as a function of $1/p$. The optimal γ and β can be found in Ref. [4, Table 4], and some examples are plotted in Figure 3. Based on the smooth pattern of the optimal γ and β up to p of 17, we guess these parameters at $p = 18, 19, 20$ using heuristics similar to that in Ref. [33]. Then evaluation of $\nu_p(\gamma, \beta)$ gives lower bounds on $\bar{\nu}_p$ at higher p which are listed in Table 2, and their corresponding γ and β are listed in Ref. [4, Table 5].

Note that, at $p = 11$ and beyond, the QAOA achieves a cut fraction better than $\frac{1}{2} + \frac{2/\pi}{\sqrt{D}}$ in the large D limit, making it the best currently known assumption-free algorithm for MaxCut on large random regular graphs.

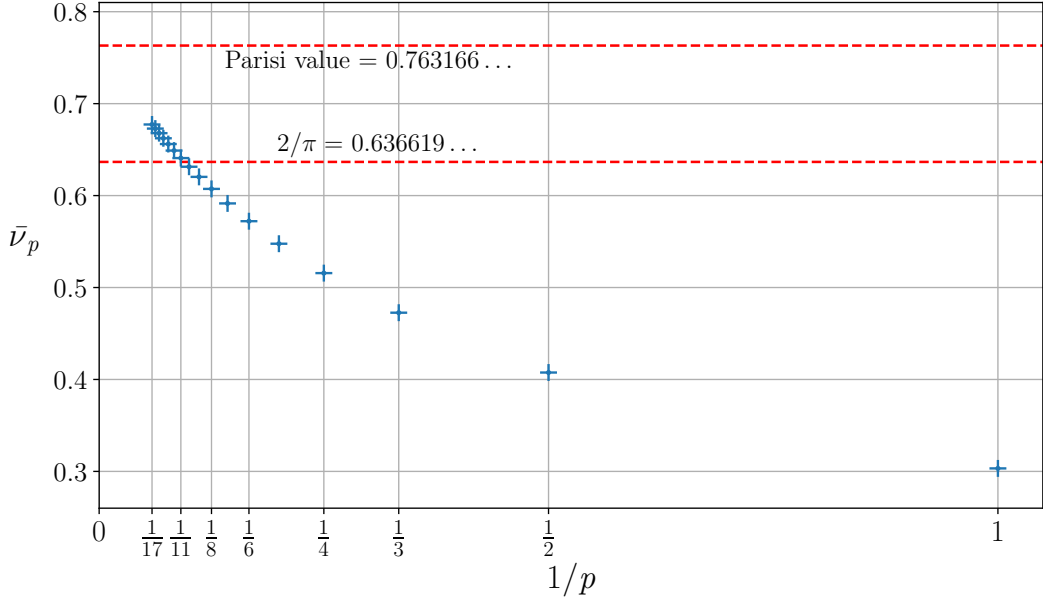
■ **Table 1** Optimal values of $\bar{\nu}_p$ up to $p = 17$.

| | | | | | | | | | |
|---------------|--------|--------|--------|--------|--------|--------|--------|--------|--------|
| p | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| $\bar{\nu}_p$ | 0.3033 | 0.4075 | 0.4726 | 0.5157 | 0.5476 | 0.5721 | 0.5915 | 0.6073 | 0.6203 |

| | | | | | | | | |
|---------------|--------|--------|--------|--------|--------|--------|--------|--------|
| p | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 |
| $\bar{\nu}_p$ | 0.6314 | 0.6408 | 0.6490 | 0.6561 | 0.6623 | 0.6679 | 0.6729 | 0.6773 |

■ **Table 2** Lower bounds of $\bar{\nu}_p$ for $p = 18, 19, 20$.

| | | | |
|---------------------------|--------|--------|--------|
| p | 18 | 19 | 20 |
| $\bar{\nu}_p$ lower bound | 0.6813 | 0.6848 | 0.6879 |



■ **Figure 2** Optimal values $\bar{\nu}_p$ as a function of $1/p$. At $p = 11$, $\bar{\nu}_p$ exceeds $2/\pi$, related to the cut fraction of the best currently known assumption-free classical algorithms. Here we made the somewhat arbitrary choice of plotting the data against $1/p$ to see the large p region in a compact plot.

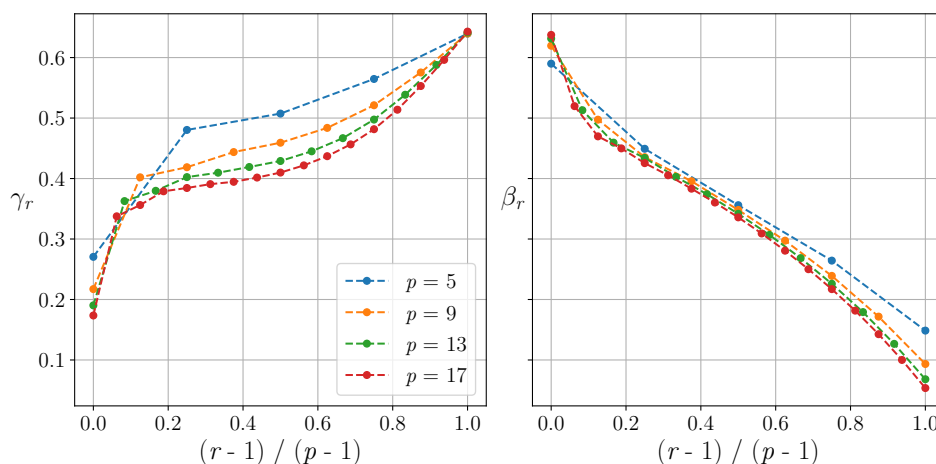
We implement the iterative procedure described in Section 3.2 in C++. Our code is available at Ref. [5]. Bit strings are encoded as `unsigned long int` variables, which allow for fast bit-wise manipulations. Matrices and vectors are implemented using the Eigen library [18]. We parallelize the sum over \mathbf{a} in Eq. (3.8) using OpenMP [11]. We optimize γ, β for each value of p using the LBFSGS++ library, which implements the Limited-memory BFGS algorithm for unconstrained optimization problems [26]. Each evaluation of the gradient of $\nu_p(\gamma, \beta)$ in Eq. (3.9) is a subroutine of the optimization which takes $2p + 1$ function calls. We run on a `n2d-highcpu-224` machine in Google Cloud, which has 224 vCPUs, using one thread per vCPU. A function call at $p = 16$ takes about 133 seconds, and a function call at $p = 17$ takes about 595 seconds. The run time of each function call is roughly multiplied by 4 every time p is increased by 1. At $p = 20$, a single function call takes slightly under 14 hours to evaluate. Memory usage is dominated by the need to store matrix $G^{(m)}$, which is negligible and quadratic in p . Further optimizations might be possible.

4 Agreement with the Sherrington-Kirkpatrick model

We note that Table 1 in this paper seems to be an extension of Table 1 in Ref. [16]. There, the authors study the performance of the QAOA on the Sherrington-Kirkpatrick (SK) model [24], which describes a spin-glass system with all-to-all random couplings. The cost function is

$$C_J^{\text{SK}}(\mathbf{z}) = \frac{1}{\sqrt{n}} \sum_{1 \leq i < j \leq n} J_{ij} z_i z_j \quad (4.1)$$

where the J_{ij} are independently drawn from a distribution with mean 0 and variance 1. The authors arrive at an iterative formula for the ensemble-averaged performance of the QAOA on the SK model



■ **Figure 3** Optimal γ_r and β_r as a function of $(r - 1)/(p - 1) \in [0, 1]$ for $p = 5, 9, 13, 17$. For each p , the index $r = 1, 2, \dots, p$ enumerates the entries of γ and β . Dashed lines in between data points are solely intended to guide the eye.

$$V_p(\gamma, \beta) := \lim_{n \rightarrow \infty} \mathbb{E}_J \left[\langle \gamma, \beta | C_J^{\text{SK}} / n | \gamma, \beta \rangle_J \right], \tag{4.2}$$

where $|\gamma, \beta\rangle_J$ is the QAOA state prepared with C_J^{SK} . Since concentration is shown to hold, we know that typical instances of the SK model all behave as the ensemble average.

Observe that $\bar{\nu}_p$, the optimized values of $\nu_p(\gamma, \beta)$, listed in Table 1 of this paper agree with the values of $\bar{V}_p = \max_{\gamma, \beta} V_p(\gamma, \beta)$ in Table 1 of Ref. [16]. It turns out that this is true in a general sense:

► **Theorem 1.** *For all p and all parameters (γ, β) , we have*

$$V_p(\gamma, \beta) = \nu_p(\gamma, \beta). \tag{4.3}$$

The proof of this theorem is provided in Ref. [4, Section 6], where the iteration for ν_p in this paper is carefully mapped to the previously known formula for V_p . This theorem establishes the fact that for each p and fixed parameters, the performance of the QAOA on large-girth D -regular graphs in the $D \rightarrow \infty$ limit is *equal* to its performance on the SK model in the $n \rightarrow \infty$ limit. We remark that in the iteration in this paper there is only one tree subgraph, with of order D^p vertices, for every large-girth D -regular graph. On the other hand, in the SK case, there is an ensemble of instances given by different weights on the complete graph. It is interesting to us that the ensemble average in Eq. (4.2) can be replaced by a single subgraph.

Theorem 1 also implies that the iteration in Section 3.2 works for evaluating the performance of the QAOA applied to both large-girth regular graphs and the SK model.

5 Conjecture that our iteration achieves the Parisi value

The cut fraction output by the QAOA on MaxCut for large-girth $(D + 1)$ -regular graphs is

$$\frac{\langle \gamma, \beta | C_{\text{MC}} | \gamma, \beta \rangle}{|E|} = \frac{1}{2} + \frac{\nu_p(D, \gamma, \beta)}{\sqrt{D}}. \tag{5.1}$$

We have given an iteration for evaluating $\nu_p(D, \gamma, \beta)$ for any depth p and parameters γ, β . Furthermore, in Section 3.2 we give a compact iteration for $\nu_p(\gamma, \beta) = \lim_{D \rightarrow \infty} \nu_p(D, \gamma, \beta)$. Using this iteration we can optimize over parameters to get $\bar{\nu}_p = \max_{\gamma, \beta} \nu_p(\gamma, \beta)$. Note $\bar{\nu}_p$ cannot be bigger than the Parisi value, $\Pi_* = \lim_{n \rightarrow \infty} \mathbb{E}_J[\max_{\mathbf{z}} C_J^{\text{SK}}(\mathbf{z})/n]$. From our numerics out to $p = 17$ we see that $\bar{\nu}_p$ is headed in that direction.

Now we make the bold conjecture:

► **Conjecture.** *Let $\Pi_* = 0.763166\dots$ be the Parisi value [25, 27]. Then*

$$\lim_{p \rightarrow \infty} \bar{\nu}_p = \Pi_*. \quad (5.2)$$

That is, the iteration in Section 3.2 is an alternative procedure to compute Π_* . To prove this conjecture, perhaps one can show that the iteration in this paper is equivalent to one of the known procedures for computing Π_* . (It may be interesting to note that $\Pi_* = \lim_{k \rightarrow \infty} \mathcal{P}_k$, where \mathcal{P}_k is the minimum of the Parisi variational principle over a k -step replica symmetry breaking ansatz with $2k + 1$ parameters [24, 2]. This is not unlike $\bar{\nu}_p$.) Or one can find a way to analytically evaluate the $p \rightarrow \infty$ limit.

There is an order of limits issue we now address. For any combinatorial optimization problem of fixed size, the QAOA can be shown to give the optimal solution in the $p \rightarrow \infty$ limit [15]. This may require p to grow exponentially in the system size. But we calculate the performance $\bar{\nu}_p$ of the QAOA at fixed p in the $D \rightarrow \infty$ limit (which means infinite system size). Then we take $p \rightarrow \infty$. Our conjecture is about whether, under this new order of limits, the QAOA achieves the optimum as $p \rightarrow \infty$.

6 Generalized iterations for Max- q -XORSAT

It turns out we can easily generalize our iterations for the QAOA's performance on MaxCut in Section 3 to the Max- q -XORSAT problem. MaxCut is a special case of Max-2-XORSAT. Given a q -uniform hypergraph $G = (V, E)$ where $E \subseteq V^q$, and given a signed weight $J_{i_1 i_2 \dots i_q} \in \{\pm 1\}$ for each edge $(i_1, i_2, \dots, i_q) \in E$, Max- q -XORSAT is the problem of maximizing the following cost function:

$$C_J^{\text{XOR}}(\mathbf{z}) = \sum_{(i_1, \dots, i_q) \in E} \frac{1}{2} (1 + J_{i_1 i_2 \dots i_q} z_{i_1} z_{i_2} \cdots z_{i_q}). \quad (6.1)$$

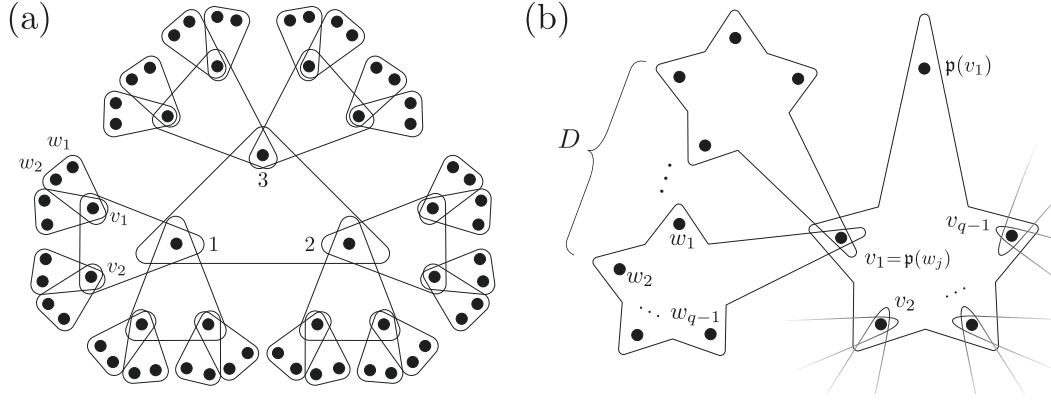
This cost function can be understood as counting the number of satisfied clauses, where a clause is satisfied if $z_{i_1} z_{i_2} \cdots z_{i_q} = J_{i_1 i_2 \dots i_q}$ on the associated edge. Note the MaxCut cost function in Eq. (2.2) is a special case of this problem where $q = 2$ and all $J_{i_1 i_2} = -1$.

We consider this problem on $(D + 1)$ -regular hypergraphs, where each vertex has degree $D + 1$, i.e., it is part of exactly $D + 1$ hyperedges. (As in Section 2, working with $(D + 1)$ -regular hypergraphs means the subgraphs that the QAOA sees are D -ary hypertrees.) The total number of hyperedges is $|E| = n(D + 1)/q$, where $n = |V|$ is the number of vertices. Due to a result by Sen [28], we know that with high probability as $n \rightarrow \infty$, the maximum fraction of satisfied clauses for a random $(D + 1)$ -regular hypergraph for sufficiently large D is

$$\frac{1}{|E|} \max_{\mathbf{z}} C_J^{\text{XOR}}(\mathbf{z}) = \frac{1}{2} + \Pi_q \sqrt{\frac{q}{2D}} + o(1/\sqrt{D}) \quad (6.2)$$

where Π_q is the generalized Parisi value that can be determined explicitly.² In particular, $\Pi_2 = \Pi_* = 0.763166\dots$

² See Ref. [28] for how this value can be calculated. Take care to note that the conventions slightly differ, and our $\Pi_q = P_q/\sqrt{2}$ where P_q is defined in Section 2.1 of Ref. [28].



■ **Figure 4** (a) The hypertree subgraph seen by the QAOA at $p = 2$ for the hyperedge $(1, 2, \dots, q)$ on a $(D + 1)$ -regular q -uniform hypergraph with girth $> 2p + 1$, for $q = 3$ and $D = 2$. (b) A partial view near the leaves of the hypertree subgraph for a general q and D . The starfish are hyperedges. Here w_1, w_2, \dots, w_{q-1} are leaf nodes in the same hyperedge, and we denote their common parent as $v_1 = \mathbf{p}(w_1) = \dots = \mathbf{p}(w_{q-1})$.

We want to evaluate how the QAOA performs on the Max- q -XORSAT problem for large-girth $(D + 1)$ -regular hypergraphs. Here, girth is defined as the minimum length of Berge cycles in the hypergraph [7]. Similar to the MaxCut problem discussed in Section 2, we will see that the QAOA has optimal parameters γ that are of order $1/\sqrt{D}$ for these graphs. For this reason, it will be convenient to prepare the QAOA state $|\gamma, \beta\rangle_J$ with the following shifted and scaled cost function operator

$$C_J = \frac{1}{\sqrt{D}} \sum_{(i_1, \dots, i_q) \in E} J_{i_1 i_2 \dots i_q} Z_{i_1} Z_{i_2} \dots Z_{i_q}. \quad (6.3)$$

For any such hypergraph, we are interested in the fraction of satisfied clauses output by the QAOA at any parameters, for any choices of $J_{i_1 i_2 \dots i_q}$ drawn from $\{+1, -1\}$. We show the following:

► **Theorem 2.** Consider C_J^{XOR} on any $(D + 1)$ -regular q -uniform hypergraphs with girth $> 2p + 1$. Let $|\gamma, \beta\rangle_J$ be the QAOA state generated using C_J . Then for any choice of J ,

$$\frac{1}{|E|} \langle \gamma, \beta | C_J^{\text{XOR}} | \gamma, \beta \rangle_J = \frac{1}{2} + \nu_p^{[q]}(D, \gamma, \beta) \sqrt{\frac{q}{2D}} \quad (6.4)$$

where $\nu_p^{[q]}(D, \gamma, \beta)$ is independent of J and can be evaluated (classically) with an iteration using $O(p4^{pq})$ time and $O(4^p)$ memory. In the infinite D limit, $\lim_{D \rightarrow \infty} \nu_p^{[q]}(D, \gamma, \beta)$ can be evaluated with an iteration using $O(p^2 4^p)$ time and $O(p^2)$ memory.

The full proof can be found in Ref. [4, Section 8], where we also describe iterations for $\nu_p^{[q]}$ in detail. It is based on the same idea as the iterations in Section 3, as we exploit the regularity of the hypertree subgraph seen by the QAOA on these hypergraphs.

In the next section, we give a part the proof that shows the J -independence $\nu_p^{[q]}$, and discuss its implication of a worst-case limitation on the QAOA's performance. In Section 6.2 that follows, we describe the infinite- D iteration and present results from its numerical evaluation.

6.1 J -independence of $\nu_p^{[q]}$ and implied worst-case limitation

We argue that the left hand side of Eq. (6.4) is independent of the choice of J 's, so there is no J needed on the right hand side. When the girth of the hypergraph is larger than $2p + 1$, the subgraph seen by the QAOA on any hyperedge is always a D -ary q -uniform hypertree. See Figure 4(a) for an example. In this figure each triangle is associated with a coupling J that can be either $+1$ or -1 . Look at the triangle containing vertices 1, 2 and 3. We can absorb the sign of J_{123} into the bit at vertex 1 as follows: if $J_{123} = -1$ do nothing, whereas if $J_{123} = +1$ flip the sign of the bit at vertex 1 by redefining $Z_1 \rightarrow -Z_1$. Then $J_{123}Z_1Z_2Z_3 \rightarrow -Z_1Z_2Z_3$ under this transformation. Now look at the triangle containing bits 1, v_1 and v_2 . The sign of $J_{1v_1v_2}$ may have been modified by the last step. But we can now absorb the sign of $J_{1v_1v_2}$ into the bit at v_1 so that $J_{1v_1v_2}Z_1Z_{v_1}Z_{v_2} \rightarrow -Z_1Z_{v_1}Z_{v_2}$. This might affect the sign of $J_{v_1w_1w_2}$ in the triangle containing v_1 , w_1 and w_2 . But we can redefine the bit at w_1 appropriately so that $J_{v_1w_1w_2}Z_{v_1}Z_{w_1}Z_{w_2} \rightarrow -Z_{v_1}Z_{w_1}Z_{w_2}$. Since there are no cycles in the hypertree, we can move through the whole picture in this way resetting all the couplings J to -1 .

We have reset all the couplings J to -1 in the picture, and we now argue that this makes the quantum expectation (6.4) independent of the J 's. At the quantum level we flip the sign of the operator Z_u by conjugating with X_u , that is, $X_uZ_uX_u = -Z_u$. Since the driver B commutes with each X_u and the initial state is an eigenstate of each X_u , we can sprinkle X_u 's into the left hand side of Eq. (6.4) and establish the J -independence of the expression coming from any particular hyperedge. Now the cost function (6.1) is a sum over the hyperedges of a given hypergraph, but the expected value of each term in the QAOA state is independent of the J 's. So for every $(D + 1)$ -regular q -uniform hypergraph with girth $> 2p + 1$ we can write

$$\frac{1}{|E|} \sum_J \langle \gamma, \beta | C_J^{\text{XOR}} | \gamma, \beta \rangle_J = \frac{1}{2} - \frac{1}{2} \langle \gamma, \beta | Z_1 Z_2 \dots Z_q | \gamma, \beta \rangle \quad (6.5)$$

where $(1, 2, 3, \dots, q)$ is any hyperedge, and the state $|\gamma, \beta\rangle$ without the J label has all the couplings set to -1 .

A corollary to this J -independence is that the QAOA at low depth fails to find the optimal assignment in the worst case. To see this, let us go back to the $q = 2$ case where we studied MaxCut on a large-girth regular graph which has all of the couplings $J = -1$. At optimal parameters, the fraction of satisfied clauses is $1/2 + \bar{\nu}_p/\sqrt{D}$ in the large D limit, where $\bar{\nu}_p \leq \Pi_*$. Consider the corresponding instance where all the couplings on the same graph are set to $J = +1$, which makes the instance fully satisfiable. In that case, the best possible fraction of satisfied clauses is 1. However, the fraction output by the QAOA at optimal parameters is the same as in the $J = -1$ case, that is, at most $1/2 + \Pi_*/\sqrt{D}$, which is only a bit more than $1/2$ in the large D limit.

Here we have an example of the QAOA failing to reach the optimum in the worst case because it does not “see” the whole graph. (Unlike previous results of the similar flavor in Refs. [9, 14], we do not need the graph to be bipartite to bound the worst-case approximation ratio.) Regardless of the signs of the couplings, the low-depth QAOA sees a tree subgraph surrounding each edge. On the tree subgraph the signs of the couplings are irrelevant so the QAOA does not distinguish between instances where the cost function favors disagreement and instances where agreement is favored. Without seeing cycles the QAOA cannot do better than what it can achieve in the most frustrated case, and this yields an upper bound on the worst-case approximation ratio.

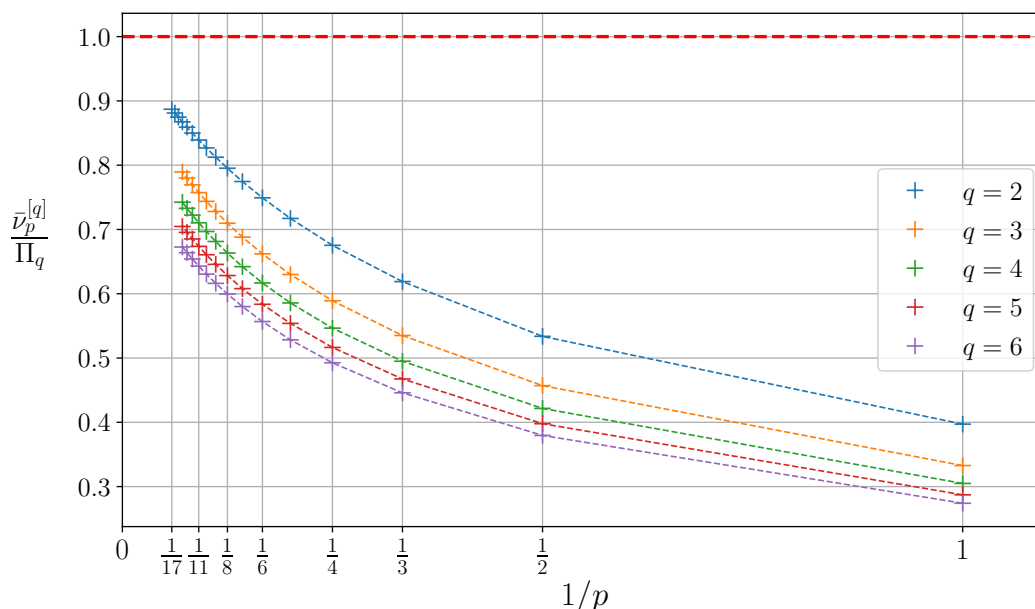
6.2 The infinite- D iteration for $\nu_p^{[q]}$

We now describe the iteration mentioned in Theorem 2 for the $D \rightarrow \infty$ limit. Similar to Section 3.2, we define matrices $G^{(m)} \in \mathbb{C}^{(2p+1) \times (2p+1)}$, for $0 \leq m \leq p$ as follows. For $j, k \in \{1, \dots, p, 0, -p, \dots, -1\}$, let $G_{j,k}^{(0)} = \sum_{\mathbf{a}} f(\mathbf{a}) a_j a_k$, and

$$G_{j,k}^{(m)} = \sum_{\mathbf{a}} f(\mathbf{a}) a_j a_k \exp \left[-\frac{1}{2} \sum_{j',k'=-p}^p (G_{j',k'}^{(m-1)})^{q-1} \Gamma_{j'} \Gamma_{k'} a_{j'} a_{k'} \right] \quad \text{for } 1 \leq m \leq p. \quad (6.6)$$

Starting at $m = 0$ and going up by p steps we arrive at $G^{(p)}$ which is used to compute

$$\nu_p^{[q]}(\gamma, \beta) := \lim_{D \rightarrow \infty} \nu_p^{[q]}(D, \gamma, \beta) = \frac{i}{\sqrt{2q}} \sum_{j=-p}^p \Gamma_j (G_{0,j}^{(p)})^q. \quad (6.7)$$

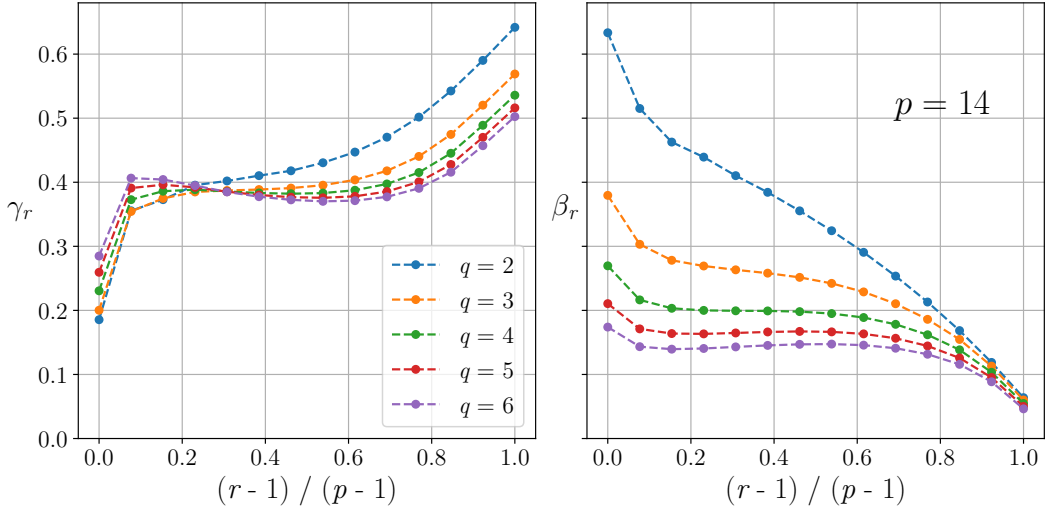


■ **Figure 5** Optimal values $\bar{\nu}_p^{[q]}$ normalized by their corresponding Parisi values Π_q as a function of $1/p$ for $q = 2, 3, 4, 5, 6$. The Parisi values are taken from Ref. [22]. Similar to Fig. 2, we made the somewhat arbitrary choice of plotting the data against $1/p$ to see the large p region in a compact plot. Dashed lines in between data points are intended to guide the eye.

Note the only difference between Max- q -XORSAT and MaxCut, where $q = 2$, can be seen by comparing Eqs. (3.8) and (3.9) in Section 3.2 to Eqs. (6.6) and (6.7) in the current iteration, where we are raising the matrix elements of G to some q -dependent power. Hence, this iteration also takes at most $O(p^2 4^p)$ time and $O(p^2)$ memory to be evaluated, regardless of q . This is polynomially faster than the finite D case with exponentially better memory usage.

We take this iteration and numerically optimize over γ and β to find

$$\bar{\nu}_p^{[q]} = \max_{\gamma, \beta} \lim_{D \rightarrow \infty} \nu_p^{[q]}(D, \gamma, \beta). \quad (6.8)$$



■ **Figure 6** Optimal QAOA parameters (γ, β) at $p = 14$ for various Max- q -XORSAT on D -regular hypergraphs in the $D \rightarrow \infty$ limit. This data can be found in Ref. [5].

up to $p = 14$ for $3 \leq q \leq 6$. Combining with the data we have for $q = 2$ in Table 1, we plot the results in Figure 5. For ease of comparison across different values of q we have normalized $\bar{\nu}_p^{[q]}$ by its corresponding Parisi value Π_q . See Figure 6 for a plot of the optimal γ and β we found at $p = 14$. Numerical values for $\bar{\nu}_p^{[q]}$ and optimal γ and β for all $1 \leq p \leq 14$ can be found in Ref. [5].

In some cases, there are thresholds on how well the QAOA at low depths can do. It is known that for problems that exhibit the overlap gap property, the locality property of the QAOA prevents it from getting close to the optimum at low depths where it does not see the whole graph [13, 10]. Specifically, using an overlap gap property in the Max- q -XORSAT problem on random Erdős-Rényi hypergraphs with constant average degree and even $q \geq 4$, Ref. [10] showed that the QAOA (or any local algorithm) has limited performance when the depth p is less than $\epsilon \log n$, where n is the graph size and ϵ is a constant that depends on the degree and q . Assuming the overlap gap property also holds when the hypergraphs are regular, one can use similar arguments to show that the QAOA's performance as measured by $\bar{\nu}_p^{[q]}/\Pi_q$ does not converge to 1 as $p \rightarrow \infty$ when $q \geq 4$ and is even. This is because our large-girth assumption implies the graph has at least D^p vertices, so p is always less than $\epsilon \log n$ in this limit.

7 Discussion

In this paper, we have introduced new techniques for evaluating the performance of a quantum algorithm at high qubit number and at high depth. In particular we do this by finding a compact iteration for the QAOA's performance on MaxCut on instances with locally tree-like neighborhoods. On random large-girth D -regular graphs, the QAOA at $p = 11$ and higher has the highest approximation ratio of any assumption-free algorithm. We have given performance guarantees for the QAOA, but it is necessary to run a quantum computer to produce a string with the calculated performance.

We have also shown that for any depth p and for any parameters, γ and β , the performance of the QAOA on large-girth D -regular graphs, as $D \rightarrow \infty$, matches the typical performance of the QAOA on the Sherrington-Kirkpatrick model at infinite size. We find it remarkable

that the ensemble averaging done in the SK model can be replaced by analyzing a single tree subgraph. For both of these models the best conceivable performance is upperbounded by the Parisi constant, Π_* . There are optimal parameters at each p , and we speculate that as $p \rightarrow \infty$ these optimal parameters give QAOA performance that matches the Parisi constant for both models.

Moreover, in Section 6, we have generalized our iteration for MaxCut on large-girth regular graphs to evaluate the QAOA's performance on Max- q -XORSAT problems for large-girth regular hypergraphs. We have shown that, at fixed parameters, the QAOA gives the same value of the objective function regardless of the signs of the couplings on these hypergraphs. This implies a worst-case algorithmic threshold at low depth for fully satisfiable instances. Building on our work, Ref. [6] recently generalized the equivalence between MaxCut and the SK model to between Max- q -XORSAT and the fully connected q -spin model.

There are a number of ideas to explore coming out of this work. Can we find a more efficient iterative formula for the QAOA's performance than the one in Section 3.2? If so, we can better probe the large- p behavior of the QAOA. Can the iteration in Section 3.2 be recast in the $p \rightarrow \infty$ limit in terms of continuous functions corresponding to γ, β ? This might be a way to verify, or falsify, the conjecture in Section 5.

Can one find other problems at high qubit number and high depth where the performance of the QAOA can be established using techniques similar to the ones introduced in this paper?

References

- 1 Ahmed El Alaoui, Andrea Montanari, and Mark Sellke. Local algorithms for Maximum Cut and Minimum Bisection on locally treelike regular graphs of large degree. *arXiv preprint*, 2021. [arXiv:2111.06813](https://arxiv.org/abs/2111.06813).
- 2 Antonio Auffinger, Wei-Kuo Chen, and Qiang Zeng. The SK Model Is Infinite Step Replica Symmetry Breaking at Zero Temperature. *Communications on Pure and Applied Mathematics*, 73(5):921–943, 2020. doi:10.1002/cpa.21886.
- 3 Boaz Barak and Kunal Marwaha. Classical Algorithms and Quantum Limitations for Maximum Cut on High-Girth Graphs. *13th Innovations in Theoretical Computer Science Conference (ITCS 2022)*, 215:14:1–14:21, 2022. doi:10.4230/LIPIcs.ITCS.2022.14.
- 4 Joao Basso, Edward Farhi, Kunal Marwaha, Benjamin Villalonga, and Leo Zhou. The Quantum Approximate Optimization Algorithm at High Depth for MaxCut on Large-Girth Regular Graphs and the Sherrington-Kirkpatrick Model. *arXiv preprint*, 2021. Full version on arXiv. [arXiv:2110.14206](https://arxiv.org/abs/2110.14206).
- 5 Joao Basso, Edward Farhi, Kunal Marwaha, Benjamin Villalonga, and Leo Zhou. Performance of the QAOA on MaxCut over Large-Girth Regular Graphs. Online, 2022. URL: <https://github.com/benjaminvillalonga/large-girth-maxcut-qaoa>.
- 6 Joao Basso, David Gamarnik, Song Mei, and Leo Zhou. Performance and limitations of the QAOA at constant levels on large sparse hypergraphs and spin glass models. *arXiv preprint*, 2022. [arXiv:2204.10306](https://arxiv.org/abs/2204.10306).
- 7 Claude Berge. *Hypergraphs, Combinatorics of Finite Sets*. North-Holland, 1989. URL: <http://compalg.inf.elte.hu/~tony/Oktatas/Algoritmusok-hatekonysaga/Berge-hypergraphs.pdf>.
- 8 Sami Boulebnane and Ashley Montanaro. Predicting parameters for the Quantum Approximate Optimization Algorithm for MAX-CUT from the infinite-size limit. *arXiv preprint*, 2021. [arXiv:2110.10685](https://arxiv.org/abs/2110.10685).
- 9 Sergey Bravyi, Alexander Kliesch, Robert Koenig, and Eugene Tang. Obstacles to variational quantum optimization from symmetry protection. *Phys. Rev. Lett.*, 125:260505, December 2020. doi:10.1103/PhysRevLett.125.260505.

- 10 Chi-Ning Chou, Peter J. Love, Juspreet Singh Sandhu, and Jonathan Shi. Limitations of Local Quantum Algorithms on Random Max-k-XOR and Beyond. *arXiv preprint*, 2021. [arXiv:2108.06049](https://arxiv.org/abs/2108.06049).
- 11 Leonardo Dagum and Ramesh Menon. OpenMP: an industry standard API for shared-memory programming. *Computational Science & Engineering, IEEE*, 5(1):46–55, 1998. doi:10.1109/99.660313.
- 12 Amir Dembo, Andrea Montanari, and Subhabrata Sen. Extremal cuts of sparse random graphs. *The Annals of Probability*, 45(2):1190–1217, 2017. doi:10.1214/15-AOP1084.
- 13 Edward Farhi, David Gamarnik, and Sam Gutmann. The Quantum Approximate Optimization Algorithm Needs to See the Whole Graph: A Typical Case. *arXiv preprint*, 2020. [arXiv:2004.09002](https://arxiv.org/abs/2004.09002).
- 14 Edward Farhi, David Gamarnik, and Sam Gutmann. The Quantum Approximate Optimization Algorithm Needs to See the Whole Graph: Worst Case Examples. *arXiv preprint*, 2020. [arXiv:2005.08747](https://arxiv.org/abs/2005.08747).
- 15 Edward Farhi, Jeffrey Goldstone, and Sam Gutmann. A Quantum Approximate Optimization Algorithm. *arXiv preprint*, 2014. [arXiv:1411.4028](https://arxiv.org/abs/1411.4028).
- 16 Edward Farhi, Jeffrey Goldstone, Sam Gutmann, and Leo Zhou. The Quantum Approximate Optimization Algorithm and the Sherrington-Kirkpatrick Model at Infinite Size. *arXiv preprint*, 2019. [arXiv:1910.08187](https://arxiv.org/abs/1910.08187).
- 17 David Gamarnik. The overlap gap property: A topological barrier to optimizing over random structures. *Proceedings of the National Academy of Sciences*, 118(41), 2021. doi:10.1073/pnas.2108492118.
- 18 Gaël Guennebaud, Benoît Jacob, et al. Eigen v3. Online, 2010. URL: <https://eigen.tuxfamily.org>.
- 19 Matthew P Harrigan, Kevin J Sung, Matthew Neeley, Kevin J Satzinger, Frank Arute, Kunal Arya, Juan Atalaya, Joseph C Bardin, Rami Barends, Sergio Boixo, et al. Quantum approximate optimization of non-planar graph problems on a planar superconducting processor. *Nature Physics*, 17(3):332–336, 2021. doi:10.1038/s41567-020-01105-y.
- 20 Russell Lyons. Factors of iid on trees. *Combinatorics, Probability and Computing*, 26(2):285–300, 2017. [arXiv:1401.4197](https://arxiv.org/abs/1401.4197).
- 21 Kunal Marwaha. Local classical MAX-CUT algorithm outperforms $p = 2$ QAOA on high-girth regular graphs. *Quantum*, 5:437, 2021. doi:10.22331/q-2021-04-20-437.
- 22 Kunal Marwaha and Stuart Hadfield. Bounds on approximating Max k XOR with quantum and classical local algorithms. *arXiv preprint*, 2021. [arXiv:2109.10833](https://arxiv.org/abs/2109.10833).
- 23 Andrea Montanari and Subhabrata Sen. Semidefinite programs on sparse random graphs and their application to community detection. In *Proceedings of the Forty-Eighth Annual ACM Symposium on Theory of Computing*, STOC '16, page 814–827, 2016. doi:10.1145/2897518.2897548.
- 24 Dmitry Panchenko. *The Sherrington-Kirkpatrick model*. Springer Science & Business Media, 2013. doi:10.1007/978-1-4614-6289-7.
- 25 Giorgio Parisi. Toward a mean field theory for spin glasses. *Physics Letters A*, 73(3):203–205, 1979. doi:10.1016/0375-9601(79)90708-4.
- 26 Yixuan Qiu and Dirk Toewe. LBFSGS++. Online, 2020. URL: <https://github.com/yixuan/LBFSGSpp>.
- 27 Manuel J Schmidt. *Replica symmetry breaking at low temperatures*. PhD thesis, Universität Würzburg, 2008. URL: <https://d-nb.info/991972910/34>.
- 28 Subhabrata Sen. Optimization on sparse random hypergraphs and spin glasses. *Random Structures & Algorithms*, 53(3):504–536, 2018. doi:10.1002/rsa.20774.
- 29 Jessica K Thompson, Ojas Parekh, and Kunal Marwaha. An explicit vector algorithm for high-girth MaxCut. *Symposium on Simplicity in Algorithms (SOSA)*, pages 238–246, 2022. doi:10.1137/1.9781611977066.17.

- 30 Luca Trevisan, Gregory B Sorkin, Madhu Sudan, and David P Williamson. Gadgets, approximation, and linear programming. *SIAM Journal on Computing*, 29(6):2074–2097, 2000. doi:10.1137/S0097539797328847.
- 31 Zhihui Wang, Stuart Hadfield, Zhang Jiang, and Eleanor G Rieffel. Quantum approximate optimization algorithm for MaxCut: A fermionic view. *Phys. Rev. A*, 97(2):022304, 2018. doi:10.1103/PhysRevA.97.022304.
- 32 Jonathan Wurtz and Peter Love. MaxCut quantum approximate optimization algorithm performance guarantees for $p > 1$. *Phys. Rev. A*, 103:042612, April 2021. doi:10.1103/PhysRevA.103.042612.
- 33 Leo Zhou, Sheng-Tao Wang, Soonwon Choi, Hannes Pichler, and Mikhail D. Lukin. Quantum approximate optimization algorithm: Performance, mechanism, and implementation on near-term devices. *Phys. Rev. X*, 10:021067, 2020. doi:10.1103/PhysRevX.10.021067.

A Proof of the iterations for MaxCut

In this appendix, we prove the correctness of the two iterations in Section 3.1 and Section 3.2. We hope this proof illustrates two key technical ideas in this paper: namely, we can exploit the regularity of the tree subgraph seen by the QAOA to yield a compact formula for its performance, and we find an algebraic simplification in the $D \rightarrow \infty$ limit. This appendix also serves as the proof of a special case of Theorem 2 at $q = 2$. The remaining proofs of our results can be found in the full version of this paper at Ref. [4].

A.1 Proof of the finite D iteration

We start by proving the finite D iteration that was stated in Section 3.1. We focus on the iteration for $p = 2$ as an example, and its generalization to other p is immediate.

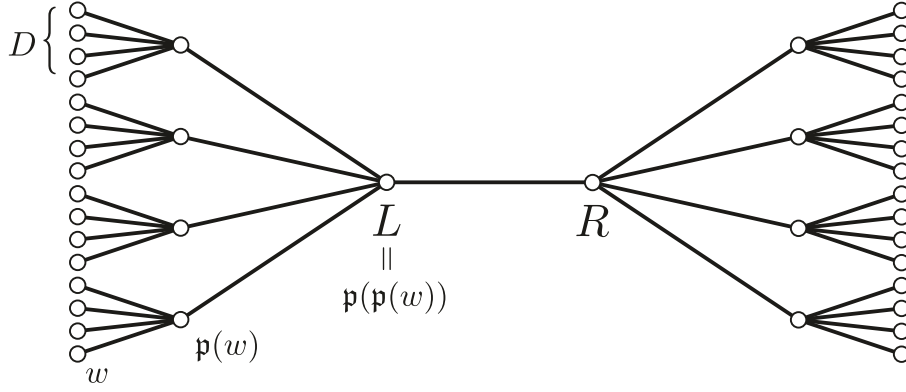
The goal is to evaluate the energy expectation for a single edge (L, R) on a $(D + 1)$ -regular graph whose girth is larger than $2p + 1$. For $p = 2$, this is

$$\langle \gamma, \beta | Z_L Z_R | \gamma, \beta \rangle = \langle s | e^{i\gamma_1 C} e^{i\beta_1 B} e^{i\gamma_2 C} e^{i\beta_2 B} Z_L Z_R e^{-i\beta_2 B} e^{-i\gamma_2 C} e^{-i\beta_1 B} e^{-i\gamma_1 C} | s \rangle \quad (\text{A.1})$$

where $C = -(1/\sqrt{D}) \sum_{(u,v) \in E} Z_u Z_v$, and E denotes the set of edges for the given graph. In the Heisenberg picture, it can be seen that the operator $e^{i\gamma_1 C} \dots e^{i\beta_p B} Z_L Z_R e^{-i\beta_p B} \dots e^{-i\gamma_1 C}$ only acts nontrivially on the subgraph induced by including all vertices distance p or less from either node L or R . For a $(D + 1)$ -regular graph with girth greater than $2p + 1$, this subgraph looks like a pair of D -ary trees that are glued at their roots (see Figure 7), with a total of $n = 2(D^p + \dots + D + 1)$ nodes. In what follows, we compute Eq. (A.1) by restricting our attention to only the qubits in this subgraph.

We start by inserting 5 complete sets in the computational Z -basis that we will label as $\mathbf{z}^{[1]}$, $\mathbf{z}^{[2]}$, $\mathbf{z}^{[0]}$, $\mathbf{z}^{[-2]}$, and $\mathbf{z}^{[-1]}$. Each of these complete sets iterates over 2^n basis states since the number of qubits in the subgraph is n . Then

$$\begin{aligned} \langle \gamma, \beta | Z_L Z_R | \gamma, \beta \rangle &= \sum_{\{\mathbf{z}^{[i]}\}} \langle s | \mathbf{z}^{[1]} \rangle e^{i\gamma_1 C(\mathbf{z}^{[1]})} \langle \mathbf{z}^{[1]} | e^{i\beta_1 B} | \mathbf{z}^{[2]} \rangle e^{i\gamma_2 C(\mathbf{z}^{[2]})} \langle \mathbf{z}^{[2]} | e^{i\beta_2 B} | \mathbf{z}^{[0]} \rangle z_L^{[0]} z_R^{[0]} \\ &\quad \times \langle \mathbf{z}^{[0]} | e^{-i\beta_2 B} | \mathbf{z}^{[-2]} \rangle e^{-i\gamma_2 C(\mathbf{z}^{[-2]})} \langle \mathbf{z}^{[-2]} | e^{-i\beta_1 B} | \mathbf{z}^{[-1]} \rangle e^{-i\gamma_1 C(\mathbf{z}^{[-1]})} \langle \mathbf{z}^{[-1]} | s \rangle \\ &= \frac{1}{2^n} \sum_{\{\mathbf{z}^{[i]}\}} \exp \left[i\gamma_1 C(\mathbf{z}^{[1]}) + i\gamma_2 C(\mathbf{z}^{[2]}) - i\gamma_2 C(\mathbf{z}^{[-2]}) - i\gamma_1 C(\mathbf{z}^{[-1]}) \right] z_L^{[0]} z_R^{[0]} \\ &\quad \times \prod_{v=1}^n \langle z_v^{[1]} | e^{i\beta_1 X} | z_v^{[2]} \rangle \langle z_v^{[2]} | e^{i\beta_2 X} | z_v^{[0]} \rangle \langle z_v^{[0]} | e^{-i\beta_2 X} | z_v^{[-2]} \rangle \langle z_v^{[-2]} | e^{-i\beta_1 X} | z_v^{[-1]} \rangle. \quad (\text{A.2}) \end{aligned}$$



■ **Figure 7** The tree subgraph seen by the QAOA at $p = 2$ for the edge (L, R) on a $(D + 1)$ -regular graph with girth $> 2p + 1$. For any node v on either of the D -ary trees we denote $\mathfrak{p}(v)$ as the parent of that node. In the figure w is a leaf node, and we show its parent and its parent's parent.

Let us define the following function which is the $p = 2$ version of Eq. (3.4):

$$f(a_1, a_2, a_0, a_{-2}, a_{-1}) = \frac{1}{2} \langle a_1 | e^{i\beta_1 X} | a_2 \rangle \langle a_2 | e^{i\beta_2 X} | a_0 \rangle \langle a_0 | e^{-i\beta_2 X} | a_{-2} \rangle \langle a_{-2} | e^{-i\beta_1 X} | a_{-1} \rangle. \quad (\text{A.3})$$

Then, using Γ as defined in Eq. (3.3), we can rewrite Eq. (A.2) as

$$\langle \gamma, \beta | Z_L Z_R | \gamma, \beta \rangle = \sum_{\{\mathbf{z}^{[i]}\}} z_L^{[0]} z_R^{[0]} \exp \left[i \sum_{j=-2}^2 \Gamma_j C(\mathbf{z}^{[j]}) \right] \prod_{v=1}^n f(\mathbf{z}_v) \quad (\text{A.4})$$

where $\mathbf{z}_v = (z_v^{[1]}, z_v^{[2]}, z_v^{[0]}, z_v^{[-2]}, z_v^{[-1]})$ are the bits from the 5 complete sets associated with node v . Using the fact that $C(\mathbf{z}) = -(1/\sqrt{D}) \sum_{(u,v) \in E} z_u z_v$, we can rewrite $\langle \gamma, \beta | Z_L Z_R | \gamma, \beta \rangle$ as

$$\langle \gamma, \beta | Z_L Z_R | \gamma, \beta \rangle = \sum_{\{\mathbf{z}_u\}} z_L^{[0]} z_R^{[0]} \exp \left[-\frac{i}{\sqrt{D}} \sum_{(u',v') \in E} \Gamma \cdot (\mathbf{z}_{u'} \mathbf{z}_{v'}) \right] \prod_{v=1}^n f(\mathbf{z}_v) \quad (\text{A.5})$$

where we have replaced the sum over the 5 complete sets $\{\mathbf{z}^{[i]} : -2 \leq i \leq 2\}$ with an equivalent sum over the bit configurations of each node $\{\mathbf{z}_u : 1 \leq u \leq n\}$. Now to evaluate $\langle Z_L Z_R \rangle$ we need to perform a sum over the bit configurations \mathbf{z}_v of every node v in the tree subgraph, where each node is coupled to its neighbors on the graph via the term in the exponential of Eq. (A.5).

We can start by considering a single leaf node w who is only connected to its parent node $\mathfrak{p}(w)$ on the tree, as shown in Figure 7. Then the sum over the 32 bit values of the configuration $\mathbf{z}_w = (z_w^{[1]}, z_w^{[2]}, z_w^{[0]}, z_w^{[-2]}, z_w^{[-1]})$ yields

$$\sum_{\mathbf{z}_w} f(\mathbf{z}_w) \exp \left[-\frac{i}{\sqrt{D}} \Gamma \cdot (\mathbf{z}_w \mathbf{z}_{\mathfrak{p}(w)}) \right] \quad (\text{A.6})$$

which is a function of the parent node's configuration $\mathbf{z}_{\mathfrak{p}(w)}$. Note that doing this on every leaf node contributes the same function to its parent. Since there are exactly D leaf nodes per parent, we get the following contribution

$$H_D^{(1)}(\mathbf{z}_{\mathbf{p}(w)}) := \left(\sum_{\mathbf{z}_w} f(\mathbf{z}_w) \exp \left[-\frac{i}{\sqrt{D}} \mathbf{\Gamma} \cdot (\mathbf{z}_w \mathbf{z}_{\mathbf{p}(w)}) \right] \right)^D. \quad (\text{A.7})$$

This is true for every parent node of any of the leaves.

After performing the sums for all the leaf nodes, we can move to the sums for their parents. Let us look at the sum on the node $\mathbf{p}(w)$ for example, which yields

$$\sum_{\mathbf{z}_{\mathbf{p}(w)}} f(\mathbf{z}_{\mathbf{p}(w)}) H_D^{(1)}(\mathbf{z}_{\mathbf{p}(w)}) \exp \left[-\frac{i}{\sqrt{D}} \mathbf{\Gamma} \cdot (\mathbf{z}_{\mathbf{p}(w)} \mathbf{z}_{\mathbf{p}(\mathbf{p}(w))}) \right]. \quad (\text{A.8})$$

Again, because its parent node $\mathbf{p}(\mathbf{p}(w))$ has D identical children like $\mathbf{p}(w)$, this yields

$$H_D^{(2)}(\mathbf{z}_{\mathbf{p}(\mathbf{p}(w))}) := \left(\sum_{\mathbf{z}_{\mathbf{p}(w)}} f(\mathbf{z}_{\mathbf{p}(w)}) H_D^{(1)}(\mathbf{z}_{\mathbf{p}(w)}) \exp \left[-\frac{i}{\sqrt{D}} \mathbf{\Gamma} \cdot (\mathbf{z}_{\mathbf{p}(w)} \mathbf{z}_{\mathbf{p}(\mathbf{p}(w))}) \right] \right)^D. \quad (\text{A.9})$$

Note at $p = 2$ we have reached the root of the tree $L = \mathbf{p}(\mathbf{p}(w))$ after these two iterations.

To evaluate $\langle \gamma, \beta | Z_L Z_R | \gamma, \beta \rangle$, it only remains to sum over the 5 bits in \mathbf{z}_L and the 5 bits in \mathbf{z}_R :

$$\langle \gamma, \beta | Z_L Z_R | \gamma, \beta \rangle = \sum_{\mathbf{z}_L, \mathbf{z}_R} z_L^{[0]} z_R^{[0]} f(\mathbf{z}_L) f(\mathbf{z}_R) H_D^{(2)}(\mathbf{z}_L) H_D^{(2)}(\mathbf{z}_R) \exp \left[-\frac{i}{\sqrt{D}} \mathbf{\Gamma} \cdot (\mathbf{z}_L \mathbf{z}_R) \right]. \quad (\text{A.10})$$

For higher p , we can see that the evaluation of $\langle \gamma, \beta | Z_L Z_R | \gamma, \beta \rangle$ simply involves more iterations of Eq. (A.9) corresponding to more levels in the tree subgraph. In summary, the iteration for general p can be written as starting with

$$H_D^{(0)}(\mathbf{a}) = 1 \quad (\text{A.11})$$

and then evaluating for $m = 1, 2, \dots, p$,

$$H_D^{(m)}(\mathbf{a}) = \left(\sum_{\mathbf{b}} f(\mathbf{b}) H_D^{(m-1)}(\mathbf{b}) \exp \left[-\frac{i}{\sqrt{D}} \mathbf{\Gamma} \cdot (\mathbf{a}\mathbf{b}) \right] \right)^D, \quad (\text{A.12})$$

since there are p levels in the tree subgraph seen by the QAOA with p layers. At the end we get

$$\langle \gamma, \beta | Z_L Z_R | \gamma, \beta \rangle = \sum_{\mathbf{a}, \mathbf{b}} a_0 b_0 f(\mathbf{a}) f(\mathbf{b}) H_D^{(p)}(\mathbf{a}) H_D^{(p)}(\mathbf{b}) \exp \left[-\frac{i}{\sqrt{D}} \mathbf{\Gamma} \cdot (\mathbf{a}\mathbf{b}) \right]. \quad (\text{A.13})$$

This is almost what we have stated for the iteration in Section 3.1.

To finish the proof, we note from Eq. (A.3) as well as its general p version in Eq. (3.4) that

$$f(-\mathbf{a}) = f(\mathbf{a}). \quad (\text{A.14})$$

We now claim that

$$H_D^{(m)}(-\mathbf{a}) = H_D^{(m)}(\mathbf{a}) \quad \text{for } 0 \leq m \leq p \quad (\text{A.15})$$

which we will show by induction on m . Note this is trivially true for the base case $m = 0$ since $H_D^{(0)}(\mathbf{a}) = 1$ is constant. Assuming that $H_D^{(m-1)}(-\mathbf{a}) = H_D^{(m-1)}(\mathbf{a})$, we can take $\mathbf{b} \rightarrow -\mathbf{b}$ in the summand of Eq. (A.12) and combine it with its original form to see that

$$H_D^{(m)}(\mathbf{a}) = \left(\sum_{\mathbf{b}} f(\mathbf{b}) H_D^{(m-1)}(\mathbf{b}) \cos \left[\frac{1}{\sqrt{D}} \boldsymbol{\Gamma} \cdot (\mathbf{a}\mathbf{b}) \right] \right)^D. \quad (\text{A.16})$$

From this form it follows that $H_D^{(m)}(-\mathbf{a}) = H_D^{(m)}(\mathbf{a})$ since \mathbf{a} only appears in the cosine which is an even function, establishing Eq. (A.15).

Similarly, we can take $\mathbf{b} \rightarrow -\mathbf{b}$ in Eq. (A.13) and combine with its original form to get

$$\langle \gamma, \beta | Z_L Z_R | \gamma, \beta \rangle = -i \sum_{\mathbf{a}, \mathbf{b}} a_0 b_0 f(\mathbf{a}) f(\mathbf{b}) H_D^{(p)}(\mathbf{a}) H_D^{(p)}(\mathbf{b}) \sin \left[\frac{1}{\sqrt{D}} \boldsymbol{\Gamma} \cdot (\mathbf{a}\mathbf{b}) \right]. \quad (\text{A.17})$$

Thus to get the ν_p as defined in Eq. (2.6) that tells us the cut fraction, we have

$$\nu_p(D, \gamma, \beta) = \frac{i\sqrt{D}}{2} \sum_{\mathbf{a}, \mathbf{b}} a_0 b_0 f(\mathbf{a}) f(\mathbf{b}) H_D^{(p)}(\mathbf{a}) H_D^{(p)}(\mathbf{b}) \sin \left[\frac{1}{\sqrt{D}} \boldsymbol{\Gamma} \cdot (\mathbf{a}\mathbf{b}) \right]. \quad (\text{A.18})$$

This proves our iteration for any finite D in Section 3.1.

A.2 Proof of $D \rightarrow \infty$ iteration

We wish to evaluate Eq. (3.7) in the $D \rightarrow \infty$ limit:

$$\lim_{D \rightarrow \infty} \nu_p(D, \gamma, \beta) = \lim_{D \rightarrow \infty} \frac{i\sqrt{D}}{2} \sum_{\mathbf{a}, \mathbf{b}} a_0 b_0 f(\mathbf{a}) f(\mathbf{b}) H_D^{(p)}(\mathbf{a}) H_D^{(p)}(\mathbf{b}) \sin \left[\frac{1}{\sqrt{D}} \boldsymbol{\Gamma} \cdot (\mathbf{a}\mathbf{b}) \right]. \quad (\text{A.19})$$

We first prove by induction that for $0 \leq m \leq p$,

$$H^{(m)}(\mathbf{a}) := \lim_{D \rightarrow \infty} H_D^{(m)}(\mathbf{a}) \quad (\text{A.20})$$

exists and is finite. For $m = 0$, our claim holds because $H_D^{(0)}(\mathbf{a}) = 1$. Assuming the claim is true for $m - 1$, we examine $H^{(m)}(\mathbf{a})$ by taking the limit on Eq. (A.16)

$$H^{(m)}(\mathbf{a}) = \lim_{D \rightarrow \infty} \left[\sum_{\mathbf{b}} f(\mathbf{b}) H_D^{(m-1)}(\mathbf{b}) \cos \left(\frac{1}{\sqrt{D}} \boldsymbol{\Gamma} \cdot (\mathbf{a}\mathbf{b}) \right) \right]^D. \quad (\text{A.21})$$

Then performing a Taylor expansion of $\cos(\dots)$, we get

$$H^{(m)}(\mathbf{a}) = \lim_{D \rightarrow \infty} \left[\sum_{\mathbf{b}} f(\mathbf{b}) H_D^{(m-1)}(\mathbf{b}) \left(1 - \frac{1}{2D} (\boldsymbol{\Gamma} \cdot (\mathbf{a}\mathbf{b}))^2 + O\left(\frac{1}{D^2}\right) \right) \right]^D. \quad (\text{A.22})$$

Using the fact that for any m ,

$$\sum_{\mathbf{a}} f(\mathbf{a}) H_D^{(m)}(\mathbf{a}) = 1 \quad (\text{A.23})$$

which is proved in Ref. [4, Lemma 5], we get

$$H^{(m)}(\mathbf{a}) = \lim_{D \rightarrow \infty} \left[1 - \frac{1}{2D} \sum_{\mathbf{b}} f(\mathbf{b}) H_D^{(m-1)}(\mathbf{b}) (\boldsymbol{\Gamma} \cdot (\mathbf{a}\mathbf{b}))^2 + O\left(\frac{1}{D^2}\right) \right]^D. \quad (\text{A.24})$$

Finally, taking the limit,

$$H^{(m)}(\mathbf{a}) = \exp \left[-\frac{1}{2} \sum_{\mathbf{b}} f(\mathbf{b}) H^{(m-1)}(\mathbf{b}) (\boldsymbol{\Gamma} \cdot (\mathbf{a}\mathbf{b}))^2 \right] \quad (\text{A.25})$$

which yields an iteration on $H^{(m)}$.

Returning to Eq. (A.19), we apply the product rule of limits to $H_D^{(p)}(\mathbf{a})$, $H_D^{(p)}(\mathbf{b})$, and $\sqrt{D} \sin[\boldsymbol{\Gamma} \cdot (\mathbf{a}\mathbf{b})/\sqrt{D}]$ and get

$$\lim_{D \rightarrow \infty} \nu_p(D, \gamma, \beta) = \frac{i}{2} \sum_{\mathbf{a}, \mathbf{b}} a_0 b_0 f(\mathbf{a}) f(\mathbf{b}) H^{(p)}(\mathbf{a}) H^{(p)}(\mathbf{b}) \boldsymbol{\Gamma} \cdot (\mathbf{a}\mathbf{b}). \quad (\text{A.26})$$

This iteration can be simplified by expanding the dot products in Eqs. (A.25) and (A.26) to get

$$H^{(m)}(\mathbf{a}) = \exp \left[-\frac{1}{2} \sum_{j,k=-p}^p \Gamma_j \Gamma_k a_j a_k \left(\sum_{\mathbf{b}} f(\mathbf{b}) H^{(m-1)}(\mathbf{b}) b_j b_k \right) \right], \quad (\text{A.27})$$

$$\lim_{D \rightarrow \infty} \nu_p(D, \gamma, \beta) = \frac{i}{2} \sum_{j=-p}^p \Gamma_j \left(\sum_{\mathbf{a}} f(\mathbf{a}) H^{(p)}(\mathbf{a}) a_0 a_j \right) \left(\sum_{\mathbf{b}} f(\mathbf{b}) H^{(p)}(\mathbf{b}) b_0 b_j \right) \quad (\text{A.28})$$

and noticing that the quantity $\sum_{\mathbf{a}} f(\mathbf{a}) H^{(m)}(\mathbf{a}) a_j a_k$ appears repeatedly. For $0 \leq m \leq p$ and $-p \leq j, k \leq p$, define

$$G_{j,k}^{(m)} := \sum_{\mathbf{a}} f(\mathbf{a}) H^{(m)}(\mathbf{a}) a_j a_k. \quad (\text{A.29})$$

For $m = 0$, this is

$$G_{j,k}^{(0)} = \sum_{\mathbf{a}} f(\mathbf{a}) a_j a_k. \quad (\text{A.30})$$

For $1 \leq m \leq p$, we plug Eq. (A.27) into Eq. (A.29) to get

$$G_{j,k}^{(m)} = \sum_{\mathbf{a}} f(\mathbf{a}) a_j a_k \exp \left[-\frac{1}{2} \sum_{j',k'=-p}^p G_{j',k'}^{(m-1)} \Gamma_{j'} \Gamma_{k'} a_{j'} a_{k'} \right]. \quad (\text{A.31})$$

Finally, Eq. (A.28) can be written as

$$\lim_{D \rightarrow \infty} \nu_p(D, \gamma, \beta) = \frac{i}{2} \sum_{j=-p}^p \Gamma_j (G_{0,j}^{(p)})^2 \quad (\text{A.32})$$

which establishes the iteration stated in Section 3.2.

A Constant Lower Bound for Any Quantum Protocol for Secure Function Evaluation

Sarah A. Osborn ✉

Virginia Polytechnic Institute and State University, Blacksburg, VA, USA

Jamie Sikora ✉

Virginia Polytechnic Institute and State University, Blacksburg, VA, USA

Abstract

Secure function evaluation is a two-party cryptographic primitive where Bob computes a function of Alice's and his respective inputs, and both hope to keep their inputs private from the other party. It has been proven that perfect (or near perfect) security is impossible, even for quantum protocols. We generalize this no-go result by exhibiting a constant lower bound on the cheating probabilities for any quantum protocol for secure function evaluation, and present many applications from oblivious transfer to the millionaire's problem. Constant lower bounds are of practical interest since they imply the impossibility to arbitrarily amplify the security of quantum protocols by any means.

2012 ACM Subject Classification Theory of computation → Cryptographic primitives; Theory of computation → Cryptographic protocols; Security and privacy → Mathematical foundations of cryptography; Theory of computation → Quantum information theory

Keywords and phrases Quantum cryptography, security analysis, secure function evaluation

Digital Object Identifier 10.4230/LIPIcs.TQC.2022.8

Related Version *Full Version:* <https://arxiv.org/abs/2203.08268>

Funding *Sarah A. Osborn:* Department of Defense Cyber Scholarship Program (DoD CySP).

1 Introduction

The first paper studying quantum cryptography was written by Stephen Wiesner in the 1970s (published in 1983) [34]. In that paper, he presented a (knowingly insecure) protocol for *multiplexing* where a receiver could choose to learn one of two bits of their choosing. Since then, this task has been referred to as 1-*out-of-2* oblivious transfer, and has been extensively studied in the quantum community [16, 29, 33, 30, 21, 11, 15, 12, 18, 20]. Indeed, since the development of quantum key distribution in 1984 [9], it has been of great interest to use quantum mechanics to develop protocols for classical tasks and push the limits of quantum theory to find optimal protocols (and their limitations).

On the other hand, it was shown in the late 1990s (and a few times since) that perfect security for a number of cryptographic tasks, including secure function evaluation, could not attain perfect, or even near perfect, security [24, 22, 23, 21, 11]. Indeed, some popular two-party cryptographic protocols, including bit commitment [14], strong coin flipping [19], oblivious transfer [15], strong die rolling [2], as well as many others, have all seen constant lower bounds presented. Constant lower bounds are of great interest for several reasons, of which we note a few. The first reason, a practical one, is that they imply that there is no way to arbitrarily amplify the security by any means (such as repeating the protocol many times and combining them in some way). The second reason, a theoretical one, now opens the question as to what are the optimal security parameters. Assuming quantum mechanics offers *some* advantage over their classical counterparts, the question now becomes to what extent is this advantage.



© Sarah A. Osborn and Jamie Sikora;
licensed under Creative Commons License CC-BY 4.0

17th Conference on the Theory of Quantum Computation, Communication and Cryptography (TQC 2022).

Editors: François Le Gall and Tomoyuki Morimae; Article No. 8; pp. 8:1–8:14

Leibniz International Proceedings in Informatics

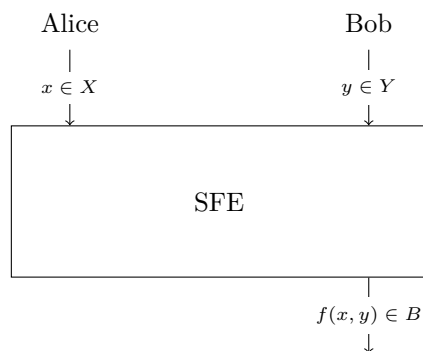


LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Note that two-party cryptography has some strange behavior, making its study very intriguing. For example, in the case of die rolling (where Alice and Bob wish to roll a die over the (possibly quantum) telephone) there can sometimes be classical protocols that offer decent security [31]. On the other hand, having classical protocols for something like coin flipping, bit commitment, and oblivious transfer is impossible [19]. And while quantum mechanics seems to deny us strong coin flipping (we have a constant lower bound [19]), it does give us arbitrarily good security for weak coin flipping [25, 3, 7, 6]. Therefore, classifying the behavior of two-party cryptographic primitives is a fruitful, and sometimes surprising, endeavor. To this end, we study the broad class of two-party cryptography known as secure function evaluation which we now discuss.

1.1 Secure function evaluation

Secure function evaluation (SFE) is a two-party cryptographic primitive in which Alice begins with an input $x \in X$ and Bob begins with an input $y \in Y$ (each input is chosen uniformly at random¹) and Bob has a deterministic function $f : X \times Y \rightarrow B$. Here, we take X , Y , and B to have finite cardinality. See Figure 1 below.



■ **Figure 1** A pictorial representation of SFE. Bob wants to compute his function f while he and Alice keep their inputs private.

The goals when designing a (quantum) protocol for SFE are:

1. *Completeness*: If both parties are honest then Bob learns f , evaluated on their inputs x and y .
2. *Soundness against cheating Bob*: Cheating Bob obtains no extra information about honest Alice's input x other than what is logically implied from knowing $f(x, y)$.
3. *Soundness against cheating Alice*: Cheating Alice obtains no information about honest Bob's input y .

It is natural to assume perfect completeness of a protocol and then to quantify the extent to which they can be made sound. In other words, we consider protocols for SFE that do what they are meant to do when Alice and Bob follow them (that is, they compute f), and then we try to find the ones which hide their respective inputs the best.

To quantify soundness against cheating Bob, for each such protocol we define the following symbols.

¹ We believe our analysis works for other probability distributions over the inputs as well, as long as they are uncorrelated. The assumption of uniformity makes certain expressions cleaner, such as the probability of Alice being able to blindly guess Bob's input.

B_{SFE} : The maximum probability with which cheating Bob can guess honest Alice's input x .

B'_{SFE} : The maximum probability with which cheating Bob can guess every $f(x, y)$, for each $y \in Y$.

Note that often these two *cheating probabilities* are the same. For instance, in 1-out-of-2 oblivious transfer we have $x \in X = \{0, 1\}^2$ as a 2-bit string, $y \in Y = \{1, 2\}$ as an index, and $f(x, y) = x_y$, i.e., the y -th bit of x . Then clearly $B_{\text{SFE}} = B'_{\text{SFE}}$, since knowing each bit is equivalent to knowing the full string. In general, $B_{\text{SFE}} \leq B'_{\text{SFE}}$, since if Bob is able to correctly learn Alice's input x , then he can compute any function of it he wants.

Similarly, to quantify soundness against cheating Alice, we define the following symbols.

A_{SFE} : The maximum probability with which cheating Alice can guess honest Bob's input y .

Note that there is only the one definition for a cheating probability for Alice since she has no output.

1.2 Main result

We now present our main result, a trade-off curve relating Alice and Bob's cheating probabilities that must be satisfied for any quantum protocol for SFE.

► **Theorem 1.** *In any quantum protocol for secure function evaluation, it holds that*

$$B'_{\text{SFE}} \geq \frac{1}{|Y| A_{\text{SFE}}} - 2(|Y| - 1) \sqrt{1 - \frac{1}{|Y| A_{\text{SFE}}}} \quad (1)$$

where Y is the set of choices for Bob's input.

We now discuss this bound. Note that

$$A_{\text{SFE}} \geq \frac{1}{|Y|}, \quad (2)$$

since she can always blindly, or randomly, guess the value of $y \in Y$. Since Alice has no output function (like Bob does) she may not be able to infer anything about y from the protocol if she is honest. Therefore, sometimes her best strategy is to randomly guess, and in this case we would have

$$A_{\text{SFE}} = \frac{1}{|Y|}, \quad (3)$$

which translates to perfect security against a cheating Alice. However, in that case, our bound implies that

$$B'_{\text{SFE}} = 1, \quad (4)$$

meaning Bob can compute his function perfectly for *every choice of input on his side*, i.e., complete insecurity against a cheating Bob. This implication exactly recovers Lo's conclusion in his 1997 paper [21], and also the conclusion in a more recent paper by Buhrman, Christandl, and Schaffner [11]. It should be mentioned that the above two papers also discuss the "Alice can cheat with a small probability" case as well. A key component in their proofs is the application of Uhlmann's theorem on purifications of the protocol to find unitaries with which Bob can use to cheat. As evidenced later on, this is very different from our proof. In fact, at no point in our protocol do we assume anything is pure and we only deal with POVMs, not

8:4 Lower Bounds on Quantum Secure Function Evaluation

unitaries. The “magic ingredient” in our proof is a generalization of Kitaev’s lower bound for strong coin flipping [19]. Moreover, we chose to quantify the security solely in terms of Alice and Bob’s cheating probabilities, which is complementary to the results in [11].

Before continuing, we now discuss what we mean by having a “constant lower bound.” To this end, we define the following symbols.

- A_{rand} : The maximum probability with which cheating Alice can guess honest Bob’s input y given only black-box access to the SFE task.
- B'_{rand} : The maximum probability with which cheating Bob can learn every $f(x, y)$, for each $y \in Y$, given only black-box access to the SFE task.

In other words, the cheating definitions above correspond to the information Alice and Bob can infer only from their outputs. Of course, Alice has no output, so clearly

$$A_{\text{rand}} = \frac{1}{|Y|}. \quad (5)$$

However, as is illustrated in our examples, it is less clear how to write B'_{rand} in terms of the parameters of a general SFE protocol.

Equipped with these symbols, we are now ready to state our constant lower bound on SFE.

► **Theorem 2.** *In any quantum protocol for secure function evaluation, either $B'_{\text{rand}} = 1$ (in which case the protocol is completely insecure), or there exists a constant $c > 1$ such that*

$$A_{\text{SFE}} \geq c \cdot A_{\text{rand}} \quad \text{or} \quad B'_{\text{SFE}} \geq c \cdot B'_{\text{rand}}. \quad (6)$$

Before discussing how to find this constant, a word on our lower bound is in order. We chose to define what it means for a constant lower bound to be a multiplicative factor. This is because A_{rand} and B'_{rand} may be dramatically different (as we demonstrate shortly). Therefore, having a constant additive factor could be unevenly weighted between cheating Alice and Bob and, we feel, would be less insightful in those cases. However, using our bound one can optimize and find an additive constant if one so desires.

To find this constant $c > 1$, note that our lower bound on B'_{SFE} (the right-hand side of Inequality (1)) is a continuous, decreasing function with respect to A_{SFE} . Therefore, if we assume

$$A_{\text{SFE}} \leq \frac{c_A}{|Y|}, \quad (7)$$

for some fixed constant $c_A \geq 1$, then we may conclude via our bound that

$$B'_{\text{SFE}} \geq \frac{1}{c_A} - 2(|Y| - 1) \sqrt{1 - \frac{1}{c_A}}. \quad (8)$$

Now, assuming that

$$B'_{\text{SFE}} = c_B \cdot B'_{\text{rand}} \quad (9)$$

for some $c_B \geq 1$, we now have the inequality

$$c_B \geq \frac{1}{B'_{\text{rand}}} \left(\frac{1}{c_A} - 2(|Y| - 1) \sqrt{1 - \frac{1}{c_A}} \right). \quad (10)$$

We shall now assume that $B'_{\text{rand}} < 1$ so that $\frac{1}{B'_{\text{rand}}} > 1$. Note that when $c_A = 1$, we have the right-hand side of (10) equalling $\frac{1}{B'_{\text{rand}}} > 1$ and when $c_A = \frac{1}{B'_{\text{rand}}}$ we have the right-hand side being strictly less than 1. Thus, by continuity of the right-hand side and the intermediate value theorem, we know there exists a constant $c > 1$ satisfying the equation

$$c = \frac{1}{B'_{\text{rand}}} \left(\frac{1}{c} - 2(|Y| - 1) \sqrt{1 - \frac{1}{c}} \right). \quad (11)$$

Note that this constant $c > 1$ is exactly what we want, since if $c_A \leq c$ then we have $c_B \geq c$.

Now, in theory one can solve for c above for a general SFE task, but it is complicated and perhaps not very insightful. However, when it comes to particular instances or families of SFE, then one can easily solve the above equation and get a constant (and possibly decent) lower bound for any quantum protocol for that task. We demonstrate this several times below.

1.3 Applications

Since our bound is general, we can apply it to many different scenarios. However, since each scenario is quite different and requires discussion, we delegate these discussions to the full version of the paper and simply summarize the cryptographic tasks below and a few of the special cases in which we found some exact formulas for lower bounds. Note that all of the special cases presented below are *new lower bounds* as far as we are aware.

- **1-out-of- n oblivious transfer.** This is where Alice has a database and Bob wishes to learn one item (his input is an index). We present lower bounds on either how much Alice can learn Bob's index or how much Bob can learn all of Alice's database. A special case is when Alice has 3 bits and Bob wants to learn 1 of them. We present a new lower bound that either

$$B_{\text{OT}} \gtrsim 0.2581 > 0.2500 \quad \text{or} \quad A_{\text{OT}} \gtrsim 0.3442 > 0.3333. \quad (12)$$

Note that we define the cheating probability symbols above in the full version, but they should be clear from context for this abbreviated discussion. This is also the case for the cheating probability symbols below.

- **k -out-of- n oblivious transfer.** This is the same as 1-out-of- n oblivious transfer except Bob's input is now a proper subset instead of an index (so Bob learns $k < n$ entries in Alice's database). We present lower bounds on either how much Alice can learn Bob's proper subset or how much Bob can learn all of Alice's database. A special case is when Alice has 4 bits and Bob wants to learn 2 of them. We present a new lower bound that either

$$B_{\text{knOT}} \gtrsim 0.2514 > 0.2500 \quad \text{or} \quad A_{\text{knOT}} \gtrsim 0.1676 > 0.1667. \quad (13)$$

- **XOR oblivious transfer.** This is similar to 1-out-of-2 oblivious transfer (where Alice's database consists of 2 bit *strings*) but Bob now has a third option of learning the bit-wise XOR of the two strings. We present lower bounds on either how much Alice can learn Bob's choice (first string, second string, or the XOR) or how much Bob can learn both of Alice's strings. A special case is when Alice's strings have length 1 (so, they are just bits). We present a new lower bound that either

$$B_{\text{XOT}} \gtrsim 0.5073 > 0.5000 \quad \text{or} \quad A_{\text{XOT}} \gtrsim 0.3382 > 0.3333. \quad (14)$$

- **Equality/one-way oblivious identification.** This is when Alice and Bob each have the same set of inputs and Bob learns whether or not their inputs are equal. We present lower bounds on either how much Alice or Bob can learn the other’s input. A special case is when the input set has cardinality 3. We present a new lower bound that either

$$B_{EQ} \gtrsim 0.671 > 0.667 \quad \text{or} \quad A_{EQ} \gtrsim 0.3355 > 0.3333. \quad (15)$$

- **Inner product.** This is when Alice and Bob each input an n -bit string and Bob learns their inner product. We present lower bounds on either how much Alice or Bob can learn the other’s input. A special case is when $n = 3$. We present a new lower bound that either

$$B_{IP} \gtrsim 0.251 > 0.250 \quad \text{or} \quad A_{IP} \gtrsim 0.1434 > 0.1429. \quad (16)$$

- **Millionaire’s problem.** This is when (rich) Alice and Bob have lots of money and Bob wishes to learn who is richer without either revealing their wealth. A special case is when $n = 10^9$ (bounding each of their bank accounts at a billion dollars). We present a new lower bound that either

$$B_{\oplus_S} \gtrsim 2 \times 10^{-9} + 5 \times 10^{-28} > 2 \times 10^{-9} \quad \text{or} \\ A_{\oplus_S} \gtrsim 1 \times 10^{-9} + 1 \times 10^{-18} + 1.25 \times 10^{-27} > 1 \times 10^{-9} + 1 \times 10^{-18} + 1 \times 10^{-27}. \quad (17)$$

We can also study a more realistic version by setting $n = 10$. We present a new lower bound that either

$$B_{\oplus_S} \gtrsim 0.2005 > 0.2000 \quad \text{or} \quad A_{\oplus_S} \gtrsim 0.1114 > 0.1111. \quad (18)$$

Therefore, some information about either Alice or Bob’s wealth is necessarily leaked.

Each of these cryptographic tasks are described further and analyzed in the full version of the paper.

1.4 Proof idea and key concepts

There are two main ingredients in proving our lower bound which we discuss at a high level below, and continue in more detail in the following sections. The magic ingredient is Kitaev’s constant lower bound for die rolling [19, 2]. Effectively what we do is use a generic SFE protocol to create a die rolling protocol, then apply Kitaev’s lower bound. However, the glue that makes SFE and die rolling play well together is a new technical result that we prove which deals with sequential gentle measurements, which we discuss next.

1.4.1 Sequential gentle measurements

The idea behind much of quantum cryptography is the concept of measurement disturbance. To put it simply, measuring to obtain certain information from a quantum state may cause it to collapse, possibly rendering it unusable for future purposes, or to simply alert honest parties that a cheating attempt was made. However, there is a concept of a *gentle measurement*, which is described at a high level below.

Gentle measurement lemma (ϵ -free version). *If a measurement outcome has a large probability of occurring, then the measured quantum state is not largely disturbed if that measurement outcome does indeed happen. (See references [36, 37] or Section 2 to see formal statements of gentle measurement lemmas.)*

How does this help us? Well, suppose for a cheating Bob who wishes to learn every $f(x, y)$, for all y , he may wish to measure some quantum state *several times*. Suppose for a fixed $y_1 \in Y$ that Bob can learn $f(x, y_1)$ with probability close to 1. Then, if he were to measure it, and achieve the correct value, then the state is not greatly disturbed, and thus more information can possibly be extracted. If a second measurement can extract the correct value of $f(x, y_2)$ for some $y_2 \in Y \setminus \{y_1\}$ with a high probability, we can repeat the process.

Now, we (intentionally) glossed over the concept of *learning*, that is, we did not precisely define it means to *learn* the correct value, in our cryptographic context. We elaborate on this in Section 2. However, it can be made precise and be put into a framework suitable for the application of a modified gentle measurement lemma. For now, we just state the main technical result of this paper below, and leave its proof for Subsection 2.2.

► **Lemma 3** (Sequential measurement lemma). *Let $f_1, \dots, f_n : X \rightarrow B$ be fixed functions and suppose Bob has a quantum encoding of $x \in X$ (where x is chosen from a probability distribution known to Bob). Suppose Bob can learn $f_i(x)$ with probability p_i for each $i \in \{1, \dots, n\}$ and let $p = \frac{1}{n} \sum_{i=1}^n p_i$ be his average success probability of learning the function values. Then Bob can learn all values $f_1(x), \dots, f_n(x)$ with probability at least*

$$p - 2(n-1)\sqrt{1-p}. \quad (19)$$

Notice that if $p \approx 1$ (meaning that Bob has a high average success probability of learning the function values) then he can learn *all* the values with probability still very close to 1. Note that this aligns with the intuition one obtains from the gentle measurement lemma. The measurement that achieves the success probability in Lemma 3 is given in Subsection 2.2.

1.4.2 Die rolling

Die rolling (DR) is a two-party cryptographic task akin to coin flipping, where Alice and Bob try to agree on a value $n \in \{0, 1, \dots, N-1\}$. The goals when designing a die rolling protocol are outlined below.

1. *Completeness*: If both parties are honest then their outcomes are uniformly random and identical.
2. *Soundness against cheating Bob*: Cheating Bob cannot influence honest Alice's outcome distribution away from uniform.
3. *Soundness against cheating Alice*: Cheating Alice cannot influence honest Bob's outcome distribution away from uniform.

For this work, we only consider perfectly complete die rolling protocols. To quantify the soundness of a die rolling protocol, we define the following symbols.

$B_{DR,n}$: The maximum probability with which cheating Bob can influence honest Alice to output the number n without Alice aborting.

$A_{DR,n}$: The maximum probability with which cheating Alice can force honest Bob to output the number n without Bob aborting.

Kitaev proved in [19] that when $N = 2$, any *quantum* protocol for die rolling satisfies

$$A_{DR,0}B_{DR,0} \geq \frac{1}{2} \quad \text{and} \quad A_{DR,1}B_{DR,1} \geq \frac{1}{2}. \quad (20)$$

Note that die rolling with $N = 2$ is simply referred to as *(strong) coin flipping* as Alice and Bob decide on one of two outcomes. Note that coin flipping is a much more studied task than die rolling, the latter being a generalization of the former. Kitaev's proof of these inequalities for coin flipping easily generalizes to similar inequalities for die rolling, namely that for any *quantum* protocol for die rolling, we have

$$A_{\text{DR},n} B_{\text{DR},n} \geq \frac{1}{N}, \text{ for all } n \in \{0, 1, \dots, N-1\}. \quad (21)$$

This is indeed a constant lower bound, as we would strive to have $A_{\text{DR},n} = B_{\text{DR},n} = \frac{1}{N}$ for all n . However, Inequality (21) implies that

$$\max\{A_{\text{DR},n}, B_{\text{DR},n}\} \geq \frac{1}{\sqrt{N}}, \text{ for all } n \in \{0, 1, \dots, N-1\} \quad (22)$$

making it impossible to get anywhere near perfect security.

1.4.3 Die rolling via secure function evaluation - gluing the two ingredients together

The first step is to create a DR protocol from a fixed SFE protocol, as shown below.

► **Protocol 4** (DR from SFE).

- Alice and Bob input uniformly random chosen inputs into a SFE protocol such that Bob learns $f(x, y)$.
- Alice selects a uniformly chosen $b \in Y$, independent from the SFE protocol. She sends b to Bob.
- Bob reveals his SFE input $y \in Y$ and also his SFE output $f(x, y)$.
- Alice computes $f(x, y)$ using x and y . If Bob's function value he sent to Alice does not match Alice's computation of the function, she aborts the protocol.
- If Alice does not abort, they both output $(b + y) \bmod |Y|$. We assume an ordering of the elements of Y is known to both Alice and Bob before the protocol, i.e., we may think of them as elements of the set $\{1, \dots, |Y|\}$.

Protocol 4 is pictorially shown in Figure 2 below.

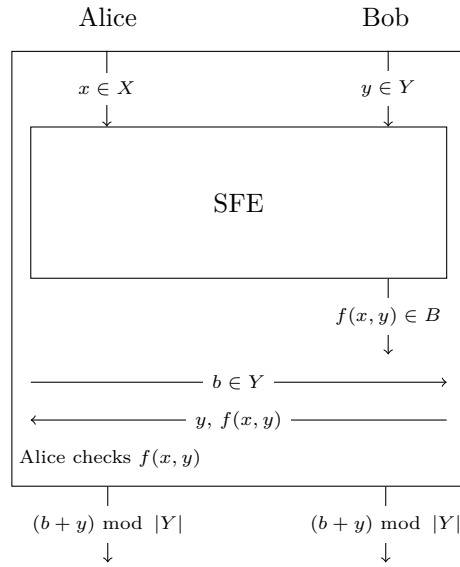
We now describe what Alice and Bob may do to cheat the die rolling protocol.

1.4.3.1 Cheating Alice

Suppose cheating Alice wants to force honest Bob to output the number 0. In this case, Alice must send b in the second to last message such that $b = y$. Since she may not know y , the probability she can successfully cheat is equal to the maximum probability with which she can learn y from the SFE protocol. However, this is precisely the definition of A_{SFE} . Thus, the case of cheating Alice is simple, we have that $A_{\text{DR},0} = A_{\text{SFE}}$.

1.4.3.2 Cheating Bob

Similar to cheating Alice, we wish to relate how much Bob can cheat in the DR protocol, say the quantity $B_{\text{DR},0}$, and how much he can cheat in the SFE protocol, namely B'_{SFE} . Suppose cheating Bob wants to force an honest Alice to output the number 0. In this case, he needs to send back y such that $y = b$ in the last message. However, for Alice to accept this last message, he must also correctly *learn* the value $f(x, y)$ from his part of the state after the



■ **Figure 2** Protocol 4: Die rolling via secure function evaluation.

SFE subroutine. In other words, before he sends his last message, he has an *encoding* of x from which he may measure to learn something. Since Alice’s message b is randomly chosen, independent of the SFE protocol, he is tasked with revealing a y with uniform probability. To say it another way, $B_{DR,0}$ is equal to the average probability that Bob is able to learn $f(x, y)$, for each y , after the SFE subroutine.

Now, to obtain a cheating strategy for Bob in SFE, consider the following. Imagine if Bob uses his optimal die rolling strategy to communicate with Alice to create the encoding of x as described above at the end of the SFE protocol. Well, we know the average success probability of Bob learning each function value; it is equal to $B_{DR,0}$, as explained above. If we now apply the sequential gentle measurement lemma, Lemma 3, we see that Bob can learn *all* the values of $f(x, y)$ with probability at least

$$B_{DR,0} - 2(|Y| - 1)\sqrt{1 - B_{DR,0}}. \tag{23}$$

Since this is a valid strategy for Bob to learn all the values of $f(x, y)$, it is a *lower bound* on B'_{SFE} .

Collecting all the above pieces of information together, and adding Kitaev’s lower bound, we have

- $A_{SFE} = A_{DR,0}$;
- $B'_{SFE} \geq B_{DR,0} - 2(|Y| - 1)\sqrt{1 - B_{DR,0}}$;
- $A_{DR,0} \cdot B_{DR,0} \geq \frac{1}{|Y|}$.

Combining these we get a proof of our main theorem, Theorem 1.

2 Learning and gentle measurements

In this section we first discuss the gentle measurement lemma and then generalize the concept to fit our needs. Then, we discuss the context in which we consider *learning* and show how to apply our generalized gentle measurement lemma.

2.1 Gentle measurements

Before we dive into gentle measurements, we must first define some essential matrix operations. Consider two matrices A and $B \in \mathbb{C}^{m \times n}$. The trace inner product is defined as

$$\langle A, B \rangle = \text{Tr}(A^* B) \quad (24)$$

where A^* represents the complex conjugate transpose of A . The trace norm of a matrix A is given by

$$\|A\|_{tr} = \text{Tr}(\sqrt{A^* A}). \quad (25)$$

The operator norm of a matrix A is given by

$$\|A\|_{op} = \sup \{ \|Av\|_2 : \|v\|_2 = 1 \} \quad (26)$$

where $\|v\|_2$ denotes the Euclidean norm $\sqrt{\langle v, v \rangle}$.

The idea behind gentle measurements is that if a measurement operator, when applied to a quantum state, produces a given result with high probability, then the post-measured state will be relatively close to the original state. For our purposes, this allows for more information to be gleaned from the state in a successive measurement. This process is formally scoped below.

► **Lemma 5** (Gentle measurement operator [36, 37]). *Consider a density operator ρ and a measurement operator Λ where $0 \leq \Lambda \leq I$. Suppose that*

$$\langle \Lambda, \rho \rangle \geq 1 - \varepsilon, \quad (27)$$

where $\varepsilon \in [0, 1]$. Then we have

$$\|\rho - \sqrt{\Lambda} \rho \sqrt{\Lambda}\|_{tr} \leq 2\sqrt{\varepsilon}. \quad (28)$$

We now use this to prove the following.

► **Lemma 6** (Sequential gentle measurement operators). *Consider a density operator ρ and measurement operators $\Lambda_1, \dots, \Lambda_n$ where $0 \leq \Lambda_k \leq I$ for each $k \in \{1, \dots, n\}$, where $n \geq 2$. Suppose that*

$$\langle \Lambda_k, \rho \rangle \geq 1 - \varepsilon_k, \quad (29)$$

where $\varepsilon_k \in [0, 1]$ for each $k \in \{1, \dots, n\}$. Then we have

$$\langle \rho, \sqrt{\Lambda_n} \cdots \sqrt{\Lambda_2} \Lambda_1 \sqrt{\Lambda_2} \cdots \sqrt{\Lambda_n} \rangle \geq 1 - \varepsilon_1 - 2 \sum_{i=2}^n \sqrt{\varepsilon_i}. \quad (30)$$

Proof. We prove this by induction.

Base case: $n = 2$. Consider the following quantity

$$|\langle \rho, \Lambda_1 \rangle - \langle \rho, \sqrt{\Lambda_2} \Lambda_1 \sqrt{\Lambda_2} \rangle| = |\langle \rho, \Lambda_1 \rangle - \langle \sqrt{\Lambda_2} \rho \sqrt{\Lambda_2}, \Lambda_1 \rangle| = |\langle \rho - \sqrt{\Lambda_2} \rho \sqrt{\Lambda_2}, \Lambda_1 \rangle|. \quad (31)$$

By applying Hölder's inequality, we get

$$|\langle \rho - \sqrt{\Lambda_2} \rho \sqrt{\Lambda_2}, \Lambda_1 \rangle| \leq \|\rho - \sqrt{\Lambda_2} \rho \sqrt{\Lambda_2}\|_{tr} \|\Lambda_1\|_{op} \leq 2\sqrt{\varepsilon_2}, \quad (32)$$

where the last inequality follows from the gentle measurement operator lemma (Lemma 5) and the assumption that $0 \leq \Lambda_1 \leq I$. This implies that

$$\langle \rho, \sqrt{\Lambda_2} \Lambda_1 \sqrt{\Lambda_2} \rangle \geq \langle \rho, \Lambda_1 \rangle - 2\sqrt{\varepsilon_2} \geq 1 - \varepsilon_1 - 2\sqrt{\varepsilon_2}. \quad (33)$$

Inductive step. Assume it is true up to some $k \in \{3, \dots, n-1\}$. We have, again, that

$$|\langle \rho - \sqrt{\Lambda_{k+1}} \rho \sqrt{\Lambda_{k+1}}, \sqrt{\Lambda_k} \cdots \sqrt{\Lambda_2} \Lambda_1 \sqrt{\Lambda_2} \cdots \sqrt{\Lambda_k} \rangle| \quad (34)$$

$$\leq \|\rho - \sqrt{\Lambda_{k+1}} \rho \sqrt{\Lambda_{k+1}}\|_{tr} \|\sqrt{\Lambda_k} \cdots \sqrt{\Lambda_2} \Lambda_1 \sqrt{\Lambda_2} \cdots \sqrt{\Lambda_k}\|_{op} \quad (35)$$

$$\leq \|\rho - \sqrt{\Lambda_{k+1}} \rho \sqrt{\Lambda_{k+1}}\|_{tr} \|\sqrt{\Lambda_k}\|_{op} \cdots \|\sqrt{\Lambda_2}\|_{op} \|\Lambda_1\|_{op} \|\sqrt{\Lambda_2}\|_{op} \cdots \|\sqrt{\Lambda_k}\|_{op} \quad (36)$$

$$\leq 2\sqrt{\varepsilon_{k+1}}, \quad (37)$$

noting that the operator norm is submultiplicative. Similar to the base case, this implies that

$$\langle \rho, \sqrt{\Lambda_{k+1}} \cdots \sqrt{\Lambda_2} \Lambda_1 \sqrt{\Lambda_2} \cdots \sqrt{\Lambda_{k+1}} \rangle \quad (38)$$

$$\geq \langle \rho, \sqrt{\Lambda_k} \cdots \sqrt{\Lambda_2} \Lambda_1 \sqrt{\Lambda_2} \cdots \sqrt{\Lambda_k} \rangle - 2\sqrt{\varepsilon_{k+1}} \quad (39)$$

$$\geq \left(1 - \varepsilon_1 - 2 \sum_{i=2}^k \sqrt{\varepsilon_i}\right) - 2\sqrt{\varepsilon_{k+1}} \quad (40)$$

$$= 1 - \varepsilon_1 - 2 \sum_{i=2}^{k+1} \sqrt{\varepsilon_i} \quad (41)$$

as desired. \blacktriangleleft

This bound is related to the quantum union bound. See [27, 17] for good versions of this bound, and also [26] for a simple proof of it. While our bound is not always stronger, it can be viewed as complementary.

2.2 Quantum encodings, and proof of Lemma 3

In this section, we pin down what it means for Bob to learn something about Alice's input.

We may assume that Alice creates the following state

$$\sum_{x \in X} p_x |x\rangle \langle x| \quad (42)$$

where p_x is the probability of her choosing x , then control all of her actions on it. That is, this is a classical register that Alice holds. After some communication, Alice and Bob will share some joint state

$$\rho := \sum_{x \in X} p_x |x\rangle \langle x| \otimes \rho_x \quad (43)$$

where ρ_x is a (quantum) encoding of Alice's bit x .

Suppose Bob wants to learn some information about x . We may assume that Alice measures her classical register in the computational basis $\{N_x : x \in X\}$ to obtain the outcome x and it is this value about which Bob wants to learn some information.

Let us assume that Bob uses the measurement $\{M_b : b \in B\}$ if he wants to learn the value of the function $f : X \rightarrow B$. In the context of SFE, this function is of the same form once a $y \in Y$ has been fixed. Now, we can calculate the probability of Bob successfully learning the function f as

$$\left\langle \rho, \sum_{x \in X} N_x \otimes M_{f(x)} \right\rangle. \quad (44)$$

Note that the structure of ρ is not really all that important, only so much as to imply that we can assume N_x is a basis measurement.

8:12 Lower Bounds on Quantum Secure Function Evaluation

Now, suppose that for a function f_i , for $i \in \{1, \dots, n\}$, Bob has a POVM $\{M_b^i : b \in B\}$ such that he learns the correct value with probability at least $1 - \varepsilon_i$. Then from the above expression, we can write

$$\left\langle \rho, \sum_{x \in X} N_x \otimes M_{f_i(x)}^i \right\rangle \geq 1 - \varepsilon_i. \quad (45)$$

By defining

$$\Lambda_i = \sum_{x \in X} N_x \otimes M_{f_i(x)}^i \quad (46)$$

we can apply Lemma 6 to get that

$$\langle \rho, \sqrt{\Lambda_n} \cdots \sqrt{\Lambda_2} \Lambda_1 \sqrt{\Lambda_2} \cdots \Lambda_n \rangle \geq 1 - \varepsilon_1 - 2 \sum_{i=2}^n \sqrt{\varepsilon_i}. \quad (47)$$

Now, the neat thing is that since $\{N_x\}$ is a basis measurement, we have that

$$\sqrt{\Lambda_n} \cdots \sqrt{\Lambda_2} \Lambda_1 \sqrt{\Lambda_2} \cdots \sqrt{\Lambda_n} = \sum_{x \in X} N_x \otimes \sqrt{M_{f_n(x)}^n} \cdots \sqrt{M_{f_2(x)}^2} M_{f_1(x)}^1 \sqrt{M_{f_2(x)}^2} \cdots \sqrt{M_{f_n(x)}^n}. \quad (48)$$

This suggests we define the POVM

$$\{\tilde{M}_{b_1, \dots, b_n} : b_1, \dots, b_n \in B\} \quad (49)$$

where

$$\tilde{M}_{b_1, \dots, b_n} := \sqrt{M_{b_n}^n} \cdots \sqrt{M_{b_2}^2} M_{b_1}^1 \sqrt{M_{b_2}^2} \cdots \sqrt{M_{b_n}^n}. \quad (50)$$

One can check that this is a valid POVM and Inequality (47) and Equation (48) show that this POVM learns $f_i(x)$ for every $i \in \{1, \dots, n\}$, with probability at least

$$1 - \varepsilon_1 - 2 \sum_{i=2}^n \sqrt{\varepsilon_i}. \quad (51)$$

Note that since the measurement operators have the POVM $\{M_b^1 : b \in B\}$ “in the middle,” and this choice was arbitrary, then we can see that Bob can create another measurement with $\{M_b^i : b \in B\}$ “in the middle” for any choice of i he wants. Thus, if he randomly chooses which measurement is “in the middle,” then we see that we can average the success probability as

$$\frac{1}{n} \sum_{j=1}^n \left(1 - \varepsilon_j - 2 \sum_{i \neq j} \sqrt{\varepsilon_i} \right) = 1 - \frac{\sum_{i=1}^n \varepsilon_i}{n} - \frac{2(n-1)}{n} \sum_{i=1}^n \sqrt{\varepsilon_i}. \quad (52)$$

Using Cauchy-Schwarz, one can prove that

$$\sum_{i=1}^n \sqrt{\varepsilon_i} \leq \sqrt{n} \sqrt{\sum_{i=1}^n \varepsilon_i}. \quad (53)$$

Therefore, the average success probability is bounded below by

$$1 - \frac{\sum_{i=1}^n \varepsilon_i}{n} - \frac{2(n-1)}{\sqrt{n}} \sqrt{\sum_{i=1}^n \varepsilon_i}. \quad (54)$$

In the context of Lemma 3, we have that $p_i = 1 - \varepsilon_i$ is the probability of guessing $f_i(x)$. Substituting this into (54), we finish our proof of Lemma 3.

References

- 1 Scott Aaronson. QMA/qpoly is contained in PSPACE/poly: de-Merlinizing quantum protocols. In *Proceedings of the 21st Annual IEEE Conference on Computational Complexity*, pages 261–273, 2006.
- 2 N Aharon and J Silman. Quantum dice rolling: a multi-outcome generalization of quantum coin flipping. *New Journal of Physics*, 12(3):033027, 2010.
- 3 Dorit Aharonov, André Chailloux, Maor Ganz, Iordanis Kerenidis, and Loïck Magnin. A simpler proof of the existence of quantum weak coin flipping with arbitrarily small bias. *SIAM Journal on Computing*, 45(3):633–679, 2016.
- 4 Andris Ambainis. A new protocol and lower bounds for quantum coin flipping. In *Proceedings of the 33rd Annual ACM Symposium on the Theory of Computing*, pages 134–142, 2001.
- 5 Andris Ambainis, Ashwin Nayak, Amnon Ta-Shma, and Umesh Vazirani. Dense quantum coding and quantum finite automata. *Journal of the ACM*, 49(4):496–511, 2002.
- 6 Atul Singh Arora, Jérémie Roland, and Chrysoula Vlachou. Analytic quantum weak coin flipping protocols with arbitrarily small bias. In *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms*, pages 919–938, 2021.
- 7 Atul Singh Arora, Jérémie Roland, and Stephan Weis. Quantum weak coin flipping. In *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing*, pages 205–216, 2019.
- 8 B. Baumgartner. An inequality for the trace of matrix products, using absolute values. Available as arXiv.org e-Print math-ph/1106.6189, 2011. [arXiv:math-ph/1106.6189](https://arxiv.org/abs/math-ph/1106.6189).
- 9 C. H. Bennett and G. Brassard. Quantum cryptography: Public key distribution and coin tossing. In *Proceedings of IEEE International Conference on Computers, Systems, and Signal Processing*, pages 175–179, India, 1984.
- 10 Manuel Blum. Coin flipping by telephone. In *Advances in Cryptology: A Report on CRYPTO 81*, pages 11–15, 1981.
- 11 Harry Buhrman, Matthias Christandl, and Christian Schaffner. Complete insecurity of quantum protocols for classical two-party computation. *Physical Review Letters*, 109(16):160501, 2012.
- 12 André Chailloux, Gus Gutoski, and Jamie Sikora. Optimal bounds for semi-honest quantum oblivious transfer. *Chicago Journal of Theoretical Computer Science*, article 13, 2016.
- 13 André Chailloux and Iordanis Kerenidis. Optimal quantum strong coin flipping. In *Proceedings of the 2009 50th Annual IEEE Symposium on Foundations of Computer Science*, pages 527–533, 2009.
- 14 André Chailloux and Iordanis Kerenidis. Optimal bounds for quantum bit commitment. In *Proceedings of the IEEE 52nd Annual Symposium on Foundations of Computer Science*, pages 354–362, 2011.
- 15 André Chailloux, Iordanis Kerenidis, and Jamie Sikora. Lower bounds for quantum oblivious transfer. *Quantum Information & Computation*, 13(1-2):158–177, 2013.
- 16 I.B. Damgaard, S. Fehr, L. Salvail, and C. Schaffner. Cryptography in the bounded quantum-storage model. In *Proceedings of the 46th Annual IEEE Symposium on Foundations of Computer Science*, pages 449–458, 2005.
- 17 Jingliang Gao. Quantum union bounds for sequential projective measurements. *Physical Review A*, 92(5):052331, 2015.
- 18 Gus Gutoski, Ansis Rosmanis, and Jamie Sikora. Fidelity of quantum strategies with applications to cryptography. *Quantum*, 2:89, 2018.
- 19 Alexei Kitaev. Quantum coin-flipping. Unpublished result, 2002. Talk at the 6th Annual workshop on Quantum Information Processing (QIP 2003).
- 20 Srijita Kundu, Jamie Sikora, and Ernest Y.-Z. Tan. A device-independent protocol for XOR oblivious transfer. In *Proceedings of the 15th Conference on the Theory of Quantum Computation, Communication and Cryptography*, volume 158(12), pages 1–15, 2020.
- 21 Hoi-Kwong Lo. Insecurity of quantum secure computations. *Physical Review A*, 56(2):1154–1162, 1997.

- 22 Hoi-Kwong Lo and H. F. Chau. Is quantum bit commitment really possible? *Physical Review Letters*, 78(17):3410–3413, 1997.
- 23 Hoi-Kwong Lo and H. F. Chau. Why quantum bit commitment and ideal quantum coin tossing are impossible. *Physica D: Nonlinear Phenomena*, 120(1-2):177–187, 1998.
- 24 Dominic Mayers. Unconditionally secure quantum bit commitment is impossible. *Physical Review Letters*, 78(17):3414–3417, 1997.
- 25 C. Mochon. Quantum weak coin flipping with arbitrarily small bias. Available as arXiv.org e-Print quant-ph/0711.4114, 2007. [arXiv:quant-ph/0711.4114](https://arxiv.org/abs/quant-ph/0711.4114).
- 26 R. O’Donnell and R. Venkateswaran. The quantum union bound made easy. Available as arXiv.org e-Print quant-ph/2103.07827, 2021. [arXiv:quant-ph/2103.07827](https://arxiv.org/abs/quant-ph/2103.07827).
- 27 Samad Khabbazi Oskouei, Stefano Mancini, and Mark M. Wilde. Union bound for quantum information processing. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 475(2221):20180612, 2018.
- 28 Michael O. Rabin. How to exchange secrets with oblivious transfer. Technical Report TR-81, Aiken Computation Lab, Harvard University, 1981.
- 29 Christian Schaffner. Cryptography in the bounded-quantum-storage model. Available as arXiv.org e-Print quant-ph/0709.0289, 2007. [arXiv:quant-ph/0709.0289](https://arxiv.org/abs/quant-ph/0709.0289).
- 30 Christian Schaffner, Barbara M. Terhal, and Stephanie Wehner. Robust cryptography in the noisy-quantum-storage model. *Quantum Information & Computation*, 9:963–996, 2009.
- 31 Jamie Sikora. Simple, near-optimal quantum protocols for die-rolling. *Cryptography*, 1(2):11, 2017.
- 32 R. W. Spekkens and T. Rudolph. Degrees of concealment and bindingness in quantum bit commitment protocols. *Physical Review A*, 65:012310, 2001.
- 33 Stephanie Wehner, Christian Schaffner, and Barbara M. Terhal. Cryptography from noisy storage. *Physical Review Letters*, 100(22):220502, 2008.
- 34 Stephen Wiesner. Conjugate coding. *SIGACT News*, 15(1):78–88, 1983.
- 35 Mark M. Wilde. Sequential decoding of a general classical-quantum channel. In *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, volume 469(2157), page 20130259, 2013.
- 36 Mark M. Wilde. *Quantum Information Theory (second edition)*. Cambridge University Press, 2017.
- 37 A. Winter. Coding theorem and strong converse for quantum channels. *IEEE Transactions on Information Theory*, 45(7):2481–2485, 1999.

Approximating Output Probabilities of Shallow Quantum Circuits Which Are Geometrically-Local in Any Fixed Dimension

Suchetan Dontha ✉

Department of Computer Science, University of Maryland, College Park, MD, USA

Shi Jie Samuel Tan ✉

Department of Computer Science, Haverford College, PA, USA

Stephen Smith ✉

Department of Mathematics, University of South Carolina, Columbia, SC, USA

Sangheon Choi ✉

Department of Computer Science, Rose-Hulman Institute of Technology, Terre Haute, IN, USA

Matthew Coudron ✉

Department of Computer Science, University of Maryland, College Park, MD, USA

Abstract

We present a classical algorithm that, for any D -dimensional geometrically-local, quantum circuit C of polylogarithmic-depth, and any bit string $x \in \{0, 1\}^n$, can compute the quantity $|\langle x|C|0^{\otimes n}\rangle|^2$ to within any inverse-polynomial additive error in quasi-polynomial time, for any fixed dimension D . This is an extension of the result [3], which originally proved this result for $D = 3$. To see why this is interesting, note that, while the $D = 1$ case of this result follows from a standard use of Matrix Product States, known for decades, the $D = 2$ case required novel and interesting techniques introduced in [1]. Extending to the case $D = 3$ was even more laborious, and required further new techniques introduced in [3]. Our work here shows that, while handling each new dimension has historically required a new insight, and fixed algorithmic primitive, based on known techniques for $D \leq 3$, we can now handle any fixed dimension $D > 3$.

Our algorithm uses the Divide-and-Conquer framework of [3] to approximate the desired quantity via several instantiations of the same problem type, each involving D -dimensional circuits on about half the number of qubits as the original. This division step is then applied recursively, until the width of the recursively decomposed circuits in the D^{th} dimension is so small that they can effectively be regarded as $(D - 1)$ -dimensional problems by absorbing the small width in the D^{th} dimension into the qudit structure at the cost of a moderate increase in runtime. The main technical challenge lies in ensuring that the more involved portions of the recursive circuit decomposition and error analysis from [3] still hold in higher dimensions, which requires small modifications to the analysis in some places. Our work also includes some simplifications, corrections and clarifications of the use of block-encodings within the original classical algorithm in [3].

2012 ACM Subject Classification Theory of computation \rightarrow Quantum complexity theory; Theory of computation \rightarrow Divide and conquer

Keywords and phrases Low-Depth Quantum Circuits, Matrix Product States, Block-Encoding

Digital Object Identifier 10.4230/LIPIcs.TQC.2022.9

Related Version *Full Version:* <https://arxiv.org/abs/2202.08349>

Funding *Suchetan Dontha:* UMD REU-CAAR.

Shi Jie Samuel Tan: UMD REU-CAAR (An Zhu-Google).

Stephen Smith: NSF-MSGI PhD summer research program.

Sangheon Choi: UMD REU-CAAR.

Matthew Coudron: NIST/QuICS.



© Suchetan Dontha, Shi Jie Samuel Tan, Stephen Smith, Sangheon Choi, and Matthew Coudron; licensed under Creative Commons License CC-BY 4.0

17th Conference on the Theory of Quantum Computation, Communication and Cryptography (TQC 2022).

Editors: François Le Gall and Tomoyuki Morimae; Article No. 9; pp. 9:1–9:17

Leibniz International Proceedings in Informatics



LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Acknowledgements MC thanks Sergey Bravyi for helpful discussions. We thank Gorjan Alagic and Nolan Coble for attending and contributing to some project group meetings. We thank the REU-CAAR program (NSF grant number 1852352) which funded SD, SJST, SC over the summer of 2021.

1 Introduction

It is known that it is $\#P$ -hard to compute the quantity $|\langle x|C|0^{\otimes n}\rangle|^2$ to within 2^{-n} additive error for low-depth, geometrically-local quantum circuits C . This fact can be deduced from early work on the hardness of simulating low-depth quantum circuits, and has a number of variations, including average-case hardness results [7, 2, 6, 5]. These hardness results indicate that computing output probabilities with such small additive error is almost certainly out of reach for both classical *and* quantum computers. If we restrict our attention to additive errors that are achievable with quantum computers, such as inverse polynomial additive error achievable by taking polynomially many samples from the quantum circuit C , then classical hardness for this estimation problem is much less clear. In fact, [1] introduced an elegant classical polynomial time algorithm for this estimation task in the case of 2D circuits. Their algorithm makes a novel use of 1D Matrix Product States carefully tailored to the 2D geometry of the circuit in question. While it is not clear how to generalize the techniques of [1] to higher dimensional circuits, [3] introduced a Divide-and-Conquer algorithm that can compute the quantity $|\langle x|C|0^{\otimes n}\rangle|^2$ to within any inverse-polynomial additive error in quasi-polynomial time for any 3D, constant-depth quantum circuit C . The algorithm in [3] works by recursively subdividing the quantum circuit C into pieces, constructed using block-encodings, and introduces new techniques for analyzing the extent to which quantum entanglement between different qubits can impact the global quantity $|\langle x|C|0^{\otimes n}\rangle|^2$.

Given the progression of ideas required to classically approximate the output probabilities of higher dimensional quantum circuits, it is natural to wonder what would be required to go even further. In this work we will show that there exists a classical quasi-polynomial time algorithm which can compute $|\langle x|C|0^{\otimes n}\rangle|^2$ to inverse polynomial additive error for any constant-depth, geometrically-local quantum circuit of fixed dimension D .

► **Theorem 1 (Main Result).** *For any D -dimensional geometrically-local, depth d quantum circuit C acting on n qubits, the algorithm $\mathcal{A}_{full}(S = (C, L, M, N), \mathcal{B}, \delta, D)$ computes the quantity $|\langle x|C|0^{\otimes n}\rangle|^2$ to within δ additive error in time $\delta^{-2} \cdot 2^{O((d \text{polylog}(n))^{D-3D})} (1/\delta)^{1/\log^2(n)}$.*¹

A key motivation for generalizing simulation results to higher dimensions exists at the level of techniques. Historically, the simulation of low-depth and geometrically local quantum circuits has required a new mathematical innovation every time the dimension, D , of the geometric locality is increased. The $D = 1$ case is solved using the famous technique of Matrix Product States (MPS), which is fundamental to the field and has been known for decades. However, it was not until recently that an algorithm was discovered for estimating output amplitudes in the case $D = 2$, and it requires a novel technique beyond standard MPS [1]. Finding an algorithm for the $D = 3$ case, [3], required a completely different approach, this time departing from the paradigm of MPS, and requiring 50 pages of mathematics to formalize

¹ For clarity we assume that the n qubits are arranged in a perfect D -dimensional cubic lattice. Here $S = (C, L, M, N)$ is the synthesis describing circuit C , as defined in this paper and in [3], and \mathcal{B} is our base-case algorithm which we specify to be the 2D algorithm of [1], and which our algorithm uses to solve subproblems which have been recursively subdivided down to 2 dimensions.

a divide-and-conquer algorithm. Our result shows that this trend of requiring completely new techniques to extend from D to $D + 1$ need not continue. One fixed divide-and-conquer algorithmic primitive, allows us to inductively establish an additive-error classical simulation algorithm for any dimension D .

Note that, while our algorithm runs in quasi-polynomial time in n for any fixed D , the runtime is triply exponential in the dimension D . If we set $D = O(\log(\log(\text{polylog}(n))))$ and δ to be inverse quasi-polynomial, then the algorithm still runs in quasi-polynomial time on a constant depth geometrically local circuit. In particular, this means that the algorithm can approximate the output probabilities of any constant depth quantum circuit that is geometrically local in $O(\log(\log(\text{polylog}(n))))$ dimensions. It is, therefore, interesting to consider the computational complexity of this problem as a function of D , since this could shed light on the extent to which arbitrary low-depth quantum circuits can be efficiently simulated. As an extreme example, an algorithm which had runtime polynomial in D could be used efficiently on constant depth quantum circuits which are not geometrically local at all. This is because any constant depth quantum circuit on n -qubits can be considered to be geometrically local in dimension $D = n$. We do not expect that our current approach can achieve a runtime polynomial in D , but we believe that even a runtime that is singly exponential in D , allowing the simulation of circuits which are geometrically local in dimension $D = \log(n)$, could have practically relevant consequences. We leave, as an open problem, the question of the optimal D -dependence for algorithms simulating constant-depth geometrically-local quantum circuits.

Our paper is organized as follows: In section 2 we review block-encodings and syntheses, both of which are used extensively throughout the algorithm. Note that section 2 primarily consists of definitions and lemmas from [3] that are tweaked for clarity and correctness. In section 3 we provide the pseudocode for our algorithm and prove our main result. The runtime and error analysis for our algorithm are located in sections 3.1 and 3.2 respectively.

2 Block-encodings and Syntheses

In order to state the pseudocode for our algorithms in Section 3 below we first need to establish a way to construct the “recursive subdivisions” of the quantum circuit C that our divide-and-conquer algorithm iteratively creates. We will concretely describe these subdivisions as “syntheses”, as defined in [3] and reviewed here for the convenience of the reader. Syntheses themselves use the idea of a block-encoding which we paraphrase below from [4].

In order to understand the following discussion, which is essential to the rest of this paper, it is necessary to read Sections 2, 3, and 4 of [3]. The lemmas repeated in this section are only included here in order to clarify or correct certain definitions in Section 3 of [3]. Many other definitions and lemmas from Sections 2, 3, and 4 of [3] are not repeated here and must be read from the original document (see the arxiv version at <https://arxiv.org/pdf/2012.05460.pdf>).

► **Definition 2** (Block-encoding). *Suppose that A is an s -qubit operator, $\alpha, \epsilon \in \mathbb{R}_+$ and $a \in \mathbb{N}$. Then we say that the $(s + a)$ -qubit unitary U is an (α, a, ϵ) -block-encoding of A , if*

$$\left\| A - \alpha(|0\rangle^{\otimes a} \otimes I)U(|0\rangle^{\otimes a} \otimes I) \right\| \leq \epsilon.$$

Consider a cut $B \cup M \cup F$ made anywhere in the cube and let $\sigma_{M \cup F} = \text{tr}_B \left(C_{B \cup M \cup F} |0\rangle\langle 0|_{B \cup M \cup F} C_{B \cup M \cup F}^\dagger \right)$. The following result is obtained by applying Lemma 45 of [4]:

9:4 Approximating Output Probabilities of Shallow Quantum Circuits

► **Lemma 3** (Block-encoding for $\sigma_{M \cup F}$). *The following is a $(1, |B \cup M \cup F|, 0)$ -block-encoding of $\sigma_{M \cup F}$:*

$$\Gamma = \left(C_{B \cup M' \cup F'}^\dagger \otimes I_{M \cup F} \right) \left(I_B \otimes \text{SWAP}_{M \cup F, M' \cup F'} \right) \left(C_{B \cup M' \cup F'} \otimes I_{M \cup F} \right).$$

In the above, $C_{B \cup M' \cup F'}$ is notation to indicate that we will be applying the circuit $C_{B \cup M \cup F}$ on the registers B , M' , and F' . In other words,

$$\sigma_{M \cup F} = (\langle 0 |_{B \cup M' \cup F'} \otimes I_{M \cup F}) \Gamma (|0 \rangle_{B \cup M' \cup F'} \otimes I_{M \cup F}).$$

The registers M' and F' above are copies of the registers M and F , respectively, and are introduced by Lemma 45 of [4]. By interleaving M' with M and B' with B and adding swap gates where appropriate, we can ensure that the resulting circuit, Γ , is still geometrically-local and has depth at most 3 times the depth of $C_{B \cup M \cup F}$. By simply moving the M register in Lemma 3 to the set of registers which are post-selected, we see that Γ is also a block-encoding of $\rho_F := \langle 0 |_M \sigma_{M \cup F} |0 \rangle_M$.

► **Lemma 4** (Block-encoding for ρ_F). *The block-encoding introduced in Lemma 3, Γ , is a $(1, |B \cup F| + 2|M|, 0)$ -block-encoding of ρ_F . Note that Lemma 4 is a correction of Lemma 7 of [3].*

Proof.

$$\begin{aligned} & (\langle 0 |_{B \cup M' \cup F' \cup M} \otimes I_F) \Gamma (|0 \rangle_{B \cup M' \cup F' \cup M} \otimes I_F) \\ &= \langle 0 |_M (\langle 0 |_{B \cup M' \cup F'} \otimes I_F) \Gamma (|0 \rangle_{B \cup M' \cup F'} \otimes I_F) |0 \rangle_M \\ &= \langle 0 |_M \sigma_{M \cup F} |0 \rangle_M \\ &= \rho_F \end{aligned} \quad \blacktriangleleft$$

Since ρ_F is the state that we are really interested in, we will henceforth refer to Γ as Γ_{ρ_F} . We can now iteratively apply Lemma 53 from [4] to obtain a block-encoding for ρ_F^k for any integer $k \geq 1$. To do this, we will need $k - 1$ copies of each of the registers B , M' , F' , and M . Let $B_1 = B$, $M'_1 = M'$, $F'_1 = F'$, and $M_1 = M$. Furthermore, for each i , $1 < i \leq k$, let B_i , M'_i , F'_i , M_i be copies of B , M' , F' , and M , respectively.

► **Lemma 5** (Block-encoding for ρ_F^k). *The following is a $(1, k|B \cup M' \cup F' \cup M|, 0)$ -block-encoding of ρ_F^k :*

$$\Gamma_{\rho_F^k} = \prod_{i=1}^k \left(C_{B_i \cup M'_i \cup F'_i}^\dagger \otimes I_{M_i \cup F} \right) \left(I_{B_i} \otimes \text{SWAP}_{M_i \cup F, M'_i \cup F'_i} \right) \left(C_{B_i \cup M'_i \cup F'_i} \otimes I_{M_i \cup F} \right).$$

In other words,

$$\rho_F^k = \left(\langle 0 |_{\mathcal{B}_k \cup \mathcal{M}'_k \cup \mathcal{F}'_k \cup \mathcal{M}_k} \otimes I_F \right) \Gamma_{\rho_F^k} \left(|0 \rangle_{\mathcal{B}_k \cup \mathcal{M}'_k \cup \mathcal{F}'_k \cup \mathcal{M}_k} \otimes I_F \right)$$

where $\mathcal{B}_k = B_1 \cup B_2 \cup \dots \cup B_k$, $\mathcal{M}'_k = M'_1 \cup M'_2 \cup \dots \cup M'_k$, etc. Note that this is a correction of equation 7 of [3]

► **Lemma 6** (Block-encoding for ρ_B^k). *Analogously, the following is a $(1, k|F \cup M' \cup B' \cup M|, 0)$ -block-encoding of ρ_B^k :*

$$\Gamma_{\rho_B^k} = \prod_{i=1}^k \left(C_{B'_i \cup M'_i \cup F_i}^\dagger \otimes I_{M_i \cup B} \right) \left(I_{F_i} \otimes \text{SWAP}_{M_i \cup B, M'_i \cup B'_i} \right) \left(C_{B'_i \cup M'_i \cup F_i} \otimes I_{M_i \cup B} \right).$$

In other words,

$$\rho_B^k = \left(\langle 0 |_{\mathcal{F}_k \cup \mathcal{M}'_k \cup \mathcal{B}'_k \cup \mathcal{M}_k} \otimes I_B \right) \Gamma_{\rho_B^k} \left(|0\rangle_{\mathcal{F}_k \cup \mathcal{M}'_k \cup \mathcal{B}'_k \cup \mathcal{M}_k} \otimes I_B \right)$$

where $\mathcal{F}_k = F_1 \cup F_2 \cup \dots \cup F_k$, $\mathcal{M}'_k = M'_1 \cup M'_2 \cup \dots \cup M'_k$, etc.

Note that this is a correction for equation 7 of [3].

Importantly, we are free to interleave all of the copies of the registers B , M and F with their originals. We do this in such a way so that we can minimally pad each 2-qubit gate from $C_{B \cup M \cup F}$ with swap gates so that this new 'padded' circuit is still geometrically local. Furthermore, the depth of this new padded circuit is at most $(2k + 1)$ times the original depth of C .

► **Definition 7 (Synthesis).** We say that an unnormalized quantum state ϕ is synthesized by a quantum circuit Γ , if Γ has three registers of qubits L, M, N such that:

$$\phi = \phi_{(\Gamma, L, M, N)} = \text{tr}_{LUM}(\langle 0_M | \Gamma | 0_{LUMUN} \rangle \langle 0_{LUMUN} | \Gamma^\dagger | 0_M \rangle). \quad (1)$$

In this case we say that the circuit Γ together with a specification of the registers L, M, N constitutes a synthesis of ϕ . When ϕ is implicit we will call this collection (Γ, L, M, N) a synthesis. This definition was taken directly from [3] and is only here for the convenience of the reader. All syntheses explicitly used in the rest of this paper are defined in section 4 of [3].

3 Algorithms and Analysis

Having discussed the essential concepts of syntheses and block-encodings in Section 2 above, we now give an explicit description of our classical simulation algorithm below. Our algorithm is divided into two pieces, Algorithm 1 and Algorithm 2. Algorithm 1 simply handles some technical edge cases for the error parameter δ , and sets the stage for making a call to Algorithm 2. Algorithm 2 contains the actual divide-and-conquer structure, describing how to perform recursive calls to itself and Algorithm 1 in one dimension lower.

The following theorem and lemmas state and prove our main result by giving runtime bounds and error bounds for Algorithm 1. Algorithm 1 is defined in complete pseudo-code below, for any dimension D , and our main result is proved by induction on dimension D .

► **Theorem 8.** For any D -dimensional geometrically-local, depth d quantum circuit C acting on n qubits, the algorithm $\mathcal{A}_{\text{full}}(S = (C, L, M, N), \mathcal{B}, \delta, D)$ computes the quantity $|\langle 0^{\otimes n} | C | 0^{\otimes n} \rangle|^2$ to within $\delta = 1/n^{\log(n)}$ additive error in time $2^{O((\text{dpolylog}(n))^{D \cdot 3^D})}$. Furthermore, let w_{D+1}, w_{D+2}, \dots be the widths of the qubit array in dimensions $D + 1, D + 2, \dots$ respectively. Then for any geometrically-local, depth d quantum circuit C acting on a lattice of n qubits having side length at most w_{D+i} in dimension $D + i$, the algorithm $\mathcal{A}_{\text{full}}(S = (C, L, M, N), \mathcal{B}, \delta, D)$ computes the quantity $|\langle 0^{\otimes n} | C | 0^{\otimes n} \rangle|^2$ to within $\delta = 1/n^{\log(n)}$ additive error in time $2^{O((\text{dpolylog}(n))^{D \cdot 3^D} w^{1/3})}$, where $w \equiv \prod_{i=1}^{\infty} w_{D+i}$.²

² We assume the n qubits are arranged in such a way that the length of each edge of the qubit lattice is $O(n^{1/D})$

Proof. We will prove Theorem 8 by induction on the dimension D . For the base-case, $D = 3$, this theorem is a direct consequence of the main result of [3]. For $D > 3$, assuming, by induction, that we have already established Theorem 8 for dimension $D - 1$, the dimension D version of the Theorem follows by Lemmas 9 and 10 respectively. The key inductive step in those two analyses happens at the point in the analysis where Algorithm 2 makes calls, such as $\mathcal{A}_{full}(S_{i,j}, \mathcal{B}, \epsilon, D - 1)$, to a $D - 1$ dimensional version of Algorithm 1. At those points the runtime and error guarantees for the $D - 1$ dimensional version of \mathcal{A}_{full} that are required by the analyses in Lemmas 9 and 10 are ensured by the inductive assumption that Theorem 8 already holds for the $D - 1$ dimensional case. \blacktriangleleft

► **Lemma 9.** *Let w be defined as in Theorem 8. Then $\mathcal{A}_{full}(S = (C, L, M, N), \mathcal{B}, \delta, D)$ runs in time $\delta^{-2} \cdot 2^{O((d \text{polylog}(n))^{D-3} w^{1/3}) (1/\delta)^{1/\log^2(n)}}$.*

Proof. The runtime analysis of \mathcal{A}_{full} begins the same as in [3]. Note that if the IF statement on Line 1 is satisfied, then the specified additive error δ is so small that we can compute the desired quantity, $|\langle 0_{ALL} | C | 0_{ALL} \rangle|^2$, exactly, by brute force, in $2^{O(n)}$ time, and this will still take less time than the guaranteed runtime:

$$T(n) = \delta^{-2} \cdot 2^{O((d \text{polylog}(n))^{D-3} w^{1/3}) (1/\delta)^{1/\log^2(n)}}.$$

Let $T_1(l, D, d, w, \delta)$ represent the run-time of algorithm 1 for a problem with side length l in dimension D with circuit depth d and thickness w in dimensions $> D$ to error δ . Let $T_2(l, D, d, w, \epsilon)$ represent the same for algorithm 2. Then we may bound T_1 as follows:

$$\begin{aligned} T_1(l, D, d, w, \delta) &< \frac{n^{1/D}}{10d} T_1(l, D - 1, d, O(wd), E_1(\delta)) + T_2(l, D, d, w, E_2(\delta)), \\ T_1(l, 2, d, w, \delta) &< \mathcal{B}(n, d, w, \delta) \end{aligned}$$

where $E_1(\delta) = 2^{\frac{\log(\delta)}{2h(n)} - 1} - 2^{\frac{\log(\delta)}{h(n)} - 1}$ and $E_2(\delta) = \delta 2^{-10 \log(n) \log(\log(n))}$.

The term $\frac{n^{1/D}}{10d} T_1(l, D - 1, d, O(wd), E_1(\delta))$ follows from lines 6-10 of algorithm 1. This entails making $\frac{n^{1/D}}{10d}$ calls of algorithm 1 on a depth d synthesis in $D - 1$ dimensions to error $E_1(\delta)$ with thickness $O(d)$ in dimension D . See the analysis of Theorem 28 of [3] for details on how this sub-problem is constructed. The term $T_2(l, D, d, w, E_2(\delta))$ refers to the call of algorithm 2 made in line 14 of algorithm 1. The base case follows directly from line 5 of algorithm 1. By standard recursion analysis, we get that

$$T_1(l, D, d, w, \delta) < n^D \mathcal{B}(n, d, O(wd^D), E_1^{(D-2)}(\delta)) + \sum_{i=0}^{D-3} n^{\frac{i}{D-i+1}} T_2(l, D - i, d, O(wd^i), E_2(E_1^{(i)}(\delta)))$$

where $E_1^{(i)}$ refers to the function E_1 composed with itself i times.

Similarly, we can bound T_2 as follows:

$$\begin{aligned} T_2(l, D, d, w, \epsilon) &< 2\Delta T_2\left(\frac{3}{4}l, D, d, w, \epsilon\right) + \Delta^2 T_1(l, D - 1, d^3 \text{polylog}(n), O(wd), \epsilon) \\ &\quad + \Delta^2 2^\Delta T_1(l, D - 1, d^3 \text{polylog}(n), O(wd), E_3(\epsilon)) \\ &\quad + 2\Delta T_1(l, D - 1, d^2 \text{polylog}(n), O(wd), \epsilon) + \text{poly}(n) \end{aligned}$$

$$T_2(O(1), D, d, w, \epsilon) = T_1(n^{1/D}, D - 1, d, O(w), \epsilon)$$

where $E_3(\epsilon) = \frac{\epsilon}{2^\Delta}$.

The term $2\Delta T_2(\frac{3}{4}l, D, d, w, \epsilon)$ follows from the calls to $\mathcal{A}(S_{L,i}, \eta - 1)$ and $\mathcal{A}(S_{i,R}, \eta - 1)$ for each i . The term $\Delta^2 T_1(l, D - 1, d^3 \text{polylog}(n), O(wd), \epsilon)$ refers to the calls to $\mathcal{A}_{full}(S_{i,j}, \mathcal{B}, \epsilon, D - 1)$ for each i and $j > i$. The term $\Delta^2 2^\Delta T_1(l, D - 1, d^3 \text{polylog}(n), O(wd), E_3(\epsilon))$ refers to the calls to

$\mathcal{A}_{full}(\left(\otimes_{k \in \sigma} \Pi_{F_k}^K \langle 0_{M_k} | \phi_{i,j} \langle \otimes_{k \in \sigma} | 0_{M_k} \rangle \Pi_{F_k}^K \right), \mathcal{B}, \frac{\epsilon}{2^\Delta}, D - 1)$ for each $i, j > i$, and σ . The term $2\Delta T_1(l, D - 1, d^2 \text{polylog}(n), O(wd), \epsilon)$ refers to the calculation of $\kappa_{T,\epsilon}$ for each i . For details regarding the construction of the sub-problems for the last three terms, refer to the run-time analysis of algorithm 2 of [3]. The final $\text{poly}(n)$ term follows from the calculation of the region Z detailed in line 8 of algorithm 2. The base case follows from the fact that if we have a problem in D dimensions with an $O(1)$ sized edge, we may apply an algorithm in $D - 1$ dimensions to solve it at the cost of an extra $O(1)$ sized thickness. By standard recursion analysis, we get that

$$\begin{aligned} T_2(l, D, d, w, \epsilon) &< (2\Delta)^\eta \cdot T_2\left(\left(\frac{3}{4}\right)^\eta l, D, d, w, \epsilon\right) \\ &+ \sum_{i=0}^{\eta-1} (2\Delta)^i (\Delta^2 T_1\left(\left(\frac{3}{4}\right)^i l, D - 1, d^3 \text{polylog}(n), O(wd), \epsilon\right) \\ &+ \Delta^2 2^\Delta T_1\left(\left(\frac{3}{4}\right)^i l, D - 1, d^3 \text{polylog}(n), O(wd), E_3(\epsilon)\right) \\ &+ 2\Delta T_1\left(\left(\frac{3}{4}\right)^i l, D - 1, d^2 \text{polylog}(n), O(wd), \epsilon\right) + \text{poly}(n)). \end{aligned}$$

Now, as we begin to substitute the recurrence relation for T_2 (in terms of T_1) into the recurrence relation for T_1 (in terms of T_2), we need to define η_i , the number of recursive calls made by T_2 to T_1 in the i -th dimension. Let us define η_i as the following:

$$\eta_i = \log_{3/4}(n^{-1/i}) = \frac{\log(n)}{i \cdot \log(4/3)}. \quad (2)$$

Now that we have defined η_i , let us substitute the T_2 recurrence relation into T_1 :

$$\begin{aligned} T_1(l, D, d, w, \delta) &< n^D \mathcal{B}(n, d, O(wd^D), E_1^{(D-2)}(\delta)) \\ &+ \sum_{i=0}^{D-3} n^{\frac{i}{D-i+1}} \left[(2\Delta)^{\eta_{D-i}} T_1\left(l, D - i - 1, d, O(wd^i), E_2(E_1^{(i)}(\delta))\right) \right. \\ &+ \sum_{j=0}^{\eta_{D-i}-1} (2\Delta)^j \left(\Delta^2 T_1\left(\left(\frac{3}{4}\right)^j l, D - i - 1, d^3 \text{polylog}(n), O(wd^{i+1}), E_2(E_1^{(i)}(\delta))\right) \right. \\ &+ \Delta^2 2^\Delta T_1\left(\left(\frac{3}{4}\right)^j l, D - i - 1, d^3 \text{polylog}(n), O(wd^{i+1}), E_3(E_2(E_1^{(i)}(\delta)))\right) \\ &\left. \left. + 2\Delta T_1\left(\left(\frac{3}{4}\right)^j l, D - i - 1, d^2 \text{polylog}(n), O(wd^{i+1}), E_2(E_1^{(i)}(\delta))\right) + \text{poly}(n) \right) \right] \end{aligned}$$

where the first T_1 term on the right-hand side comes from unrolling the T_2 term in T_2 's recurrence relation down to its base-case.

We can then continue to simplify the upper bound by combining the three terms in the second summation term into $3\Delta^2 2^\Delta T_1\left(\left(\frac{3}{4}\right)^j l, D - i - 1, d^3 \text{polylog}(n), O(wd^{i+1}), E_3(E_2(E_1^{(i)}(\delta)))\right)$ since $3\Delta^2 2^\Delta \geq (2\Delta + \Delta^2 2^\Delta + \Delta^2)$ and $E_3(E_2(E_1^{(i)}(\delta))) \leq E_2(E_1^{(i)}(\delta))$.

$$\begin{aligned}
T_1(l, D, d, w, \delta) &< n^D \mathcal{B}(n, d, O(wd^D), E_1^{(D-2)}(\delta)) \\
&+ \sum_{i=0}^{D-3} n^{\frac{i}{D-i+1}} \left[(2\Delta)^{\eta_{D-i}} T_1 \left(l, D-i-1, d, O(wd^i), E_2(E_1^{(i)}(\delta)) \right) \right. \\
&\left. + \sum_{j=0}^{\eta_{D-i}-1} (2\Delta)^j \left(3\Delta^2 2^\Delta T_1 \left(\left(\frac{3}{4} \right)^j l, D-i-1, d^3 \text{polylog}(n), O(wd^{i+1}), E_3(E_2(E_1^{(i)}(\delta))) \right) \right) \right]
\end{aligned}$$

Next, we can unpack the bracket in the first summation term to get the following:

$$\begin{aligned}
T_1(l, D, d, w, \delta) &< n^D \mathcal{B}(n, d, O(wd^D), E_1^{(D-2)}(\delta)) \\
&+ \sum_{i=0}^{D-3} n^{\frac{i}{D-i+1}} (2\Delta)^{\eta_{D-i}} T_1 \left(l, D-i-1, d, O(wd^i), E_2(E_1^{(i)}(\delta)) \right) \\
&+ \sum_{i=0}^{D-3} \sum_{j=0}^{\eta_{D-i}-1} (2\Delta)^j n^{\frac{i}{D-i+1}} \left(3\Delta^2 2^\Delta T_1 \left(\left(\frac{3}{4} \right)^j l, D-i-1, d^3 \text{polylog}(n), \right. \right. \\
&\quad \left. \left. O(wd^{i+1}), E_3(E_2(E_1^{(i)}(\delta))) \right) \right)
\end{aligned}$$

The following expression can be obtained by extracting the T_1 terms from the summation terms. We do that by bounding all the T_1 terms in the first summation term by $T_1 \left(l, D-1, d, O(wd^D), E_2(E_1^{(D)}(\delta)) \right)$ since the runtime will be longer when we start on higher dimension D instead of dimension $D-i-1$, larger thickness $O(wd^D)$ instead of thickness $O(wd^i)$, and smaller error $E_2(E_1^{(D)}(\delta))$ instead of $E_2(E_1^{(i)}(\delta))$. A similar argument could be made for the T_1 terms in the second summation term.

$$\begin{aligned}
T_1(l, D, d, w, \delta) &< n^D \mathcal{B}(n, d, O(wd^D), E_1^{(D-2)}(\delta)) \\
&+ T_1 \left(l, D-1, d, O(wd^D), E_2(E_1^{(D)}(\delta)) \right) \sum_{i=0}^{D-3} n^{\frac{i}{D-i+1}} (2\Delta)^{\eta_{D-i}} \\
&+ 3\Delta^2 2^\Delta T_1 \left(l, D-1, d^3 \text{polylog}(n), O(wd^D), E_3(E_2(E_1^{(D)}(\delta))) \right) \sum_{i=0}^{D-3} \sum_{j=0}^{\eta_{D-i}-1} (2\Delta)^j n^{\frac{i}{D-i+1}}
\end{aligned}$$

In the following step, for the first summation term, we bound the $n^{\frac{i}{D-i+1}}$ term by $\text{poly}(n)$ and the $(2\Delta)^{\eta_{D-i}}$ term by $2^{\text{polylog}(n)}$ since $\Delta, \eta = O(\log(n))$. Since we have $O(D)$ terms in the first summation term, we get $O(D \text{poly}(n) 2^{\text{polylog}(n)})$. Likewise, we can do the same thing for the second summation term to get the same upper bound.

$$\begin{aligned}
T_1(l, D, d, w, \delta) &< n^D \mathcal{B}(n, d, O(wd^D), E_1^{(D-2)}(\delta)) \\
&+ T_1 \left(l, D-1, d, O(wd^D), E_2(E_1^{(D)}(\delta)) \right) O(D \text{poly}(n) 2^{\text{polylog}(n)}) \\
&+ 3\Delta^2 2^\Delta T_1 \left(l, D-1, d^3 \text{polylog}(n), O(wd^D), E_3(E_2(E_1^{(D)}(\delta))) \right) O(D \text{poly}(n) 2^{\text{polylog}(n)}).
\end{aligned}$$

The following expression can be obtained by combining the second and third term in the previous expression. We get $T_1 \left(l, D-1, d^3 \text{polylog}(n), O(wd^D), E_3(E_2(E_1^{(D)}(\delta))) \right)$ since $d^3 \text{polylog}(n) > d$ and $E_3(E_2(E_1^{(i)}(\delta))) \leq E_2(E_1^{(i)}(\delta))$ which would give us a larger runtime bound. The $3\Delta^2 2^\Delta$ can be absorbed into the $O(D \text{poly}(n) 2^{\text{polylog}(n)})$ term.

$$T_1(l, D, d, w, \delta) < n^D \mathcal{B}(n, d, O(wd^D), E_1^{(D-2)}(\delta)) \\ + T_1\left(l, D-1, d^3 \text{polylog}(n), O(wd^D), E_3(E_2(E_1^{(D)}(\delta)))\right) O(D \text{poly}(n) 2^{\text{polylog}(n)})$$

Now, we substitute the BGM algorithm's runtime from Theorem 5 of [1] into the first term to get the recurrence for T_1 in dimension D in terms of T_1 in one dimension lower.

$$T_1(l, D, d, w, \delta) < \text{poly}(n^D) (E_1^{(D-2)}(\delta))^{-2} 2^{d^3 w^{1/D}} \\ + T_1\left(l, D-1, d^3 \text{polylog}(n), O(wd^D), E_3(E_2(E_1^{(D)}(\delta)))\right) O(D \text{poly}(n) 2^{\text{polylog}(n)})$$

Before we begin to unroll the recurrence relation for T_1 with respect to its dimension, let us first define $f(d)$, the depth of the block-encoding (at this point in the analysis), and $f^{(k)}(d)$, the depth of the block-encoding after unrolling the recurrence relation for k dimensions

$$f(d) = d^3 \text{polylog}(n) \\ f^{(k)}(d) < d^{3^k} (\text{polylog}(n))^{3^k}$$

We can also define $g(d, w)$, the thickness of the circuit (at this point in the analysis), and $g^{(k)}(d)$, the thickness of the circuit after unrolling the recurrence relation for k dimensions as follows:

$$g(d, w) = O(wd^D) \\ g^{(k)}(d, w) = g^{(k-1)}(d, w) (f^{(k-1)}(d))^D \\ = g^{(k-2)}(d, w) (f^{(k-2)}(d))^D (f^{(k-1)}(d))^D \\ = g^{(k-\ell)}(d, w) \prod_{i=1}^{\ell} (f^{(k-i)}(d))^D \\ = g(d, w) \prod_{i=1}^{k-1} (f^{(k-i)}(d))^D \\ < O(wd^D) \left(\prod_{i=1}^{k-1} (d \text{polylog}(n))^{3^i} \right)^D \\ < O(wd^D) (d \text{polylog}(n))^{D 3^k}.$$

Now, we want to write the unrolling of the recurrence relation for T_1 with respect to dimensions in terms of $f(d)$ and $g(d, w)$. To simplify the writing, we define $E_5(\delta) = E_3(E_2(E_1^{(D)}(\delta)))$. The following expression is obtained by unrolling T_1 's recurrence relation for an arbitrary dimension D to dimension 2 which is the base-case for T_1 .

$$T_1(l, D, d, w, \delta) < O((D \text{poly}(n^D) 2^{\text{polylog}(n)})^{D-2}) T_1(l, 2, f^{(D-2)}(d), g^{(D-2)}(d, w), E_5^{(D-2)}(\delta)) \\ + \sum_{i=0}^{D-3} O((D \text{poly}(n^D) 2^{\text{polylog}(n)})^i) \text{poly}(n) \left(E_1^{(D-i-2)} \left(E_5^{(i)}(\delta) \right) \right)^{-2} 2^{(f^{(i)}(d))^3 (g^{(i)}(d, w))^{\frac{1}{D-i}}}.$$

To further simplify, we replace each occurrence of i in order to maximize each quantity, then replace each occurrence of $D-2$ with D . Note that the more we compose E_1 and E_5 , the smaller they get and hence their inverse-squared form will be larger. For $f(d)$ and

9:10 Approximating Output Probabilities of Shallow Quantum Circuits

$g(d, w)$, the more we composed them, the greater the depth and the thicker the thickness of the block-encoding which gives us an upper bound for the runtime. Note how we chose the upper bound for the exponent of the $g^{(D)}(d, w)$ to be $\frac{1}{3}$ to get the smallest root form to maximize the exponent of the 2 term.

$$T_1(\ell, D, d, w, \delta) < O((D \text{poly}(n) 2^{\text{polylog}(n^D)})^D) T_1(\ell, 2, f^{(D)}(d), g^{(D)}(d, w), E_5^{(D)}(\delta)) \\ + D \cdot O((D \text{poly}(n^D) 2^{\text{polylog}(n)^D}) \text{poly}(n) \left(E_1^{(D)} \left(E_5^{(D)}(\delta) \right) \right)^{-2} 2^{(f^{(D)}(d))^3 (g^{(D)}(d, w))^{\frac{1}{3}}})$$

Next, we write the first term of the right-hand side of the first inequality according to BGM's runtime as given in Theorem 5 of [1] and then brought the $D \cdot \text{poly}(n)$ coefficient in the second term into the second term's big- O . The second inequality comes from the fact that the first term is smaller than the second term and hence can be absorbed into the second term.

$$T_1(\ell, D, d, w, \delta) < O((\text{poly}(n^D))^{D+1} (D 2^{\text{polylog}(n)^D}) (E_5^{(D)}(\delta))^{-2} 2^{(f^{(D)}(d))^2 (g^{(D)}(d, w))^{1/D}} \\ + O((D \text{poly}(n^D))^{D+1} (2^{\text{polylog}(n)^D}) \left(E_1^{(D)} \left(E_5^{(D)}(\delta) \right) \right)^{-2} 2^{(f^{(D)}(d))^3 (g^{(D)}(d, w))^{\frac{1}{3}}}) \\ < O((D \text{poly}(n^D))^{D+1} (2^{\text{polylog}(n)^D}) \left(E_1^{(D)} \left(E_5^{(D)}(\delta) \right) \right)^{-2} 2^{(f^{(D)}(d))^3 (g^{(D)}(d, w))^{\frac{1}{3}}})$$

Now, we substitute the upper bounds for $f^{(k)}(d)$ and $g^{(k)}(d, w)$ as previously defined into the above expression to get the following inequality:

$$T_1(\ell, D, d, w, \delta) < O((D \text{poly}(n^D))^{D+1} (2^{\text{polylog}(n)^D}) \left(E_1^{(D)} \left(E_5^{(D)}(\delta) \right) \right)^{-2} \\ \cdot 2^{(d \text{polylog}(n))^3 D+1} (O(w d^D)^{\frac{1}{3}} (d \text{polylog}(n))^{D 3^{D-1}}) \\ = O((D \text{poly}(n^D))^{D+1} (2^{\text{polylog}(n)^D}) \left(E_1^{(D)} \left(E_5^{(D)}(\delta) \right) \right)^{-2} \\ \cdot 2^{O(w d^D)^{\frac{1}{3}} (d \text{polylog}(n))^{(9+D) 3^{D-1}}} \\ < O((D \text{poly}(n^D))^{D+1} (2^{\text{polylog}(n)^D}) \left(E_1^{(D)} \left(E_5^{(D)}(\delta) \right) \right)^{-2} \\ \cdot 2^{O(d^{(3^{D+1}+D/3+D 3^{D-1})} (\text{polylog}(n))^{(3^{D+1}+D 3^{D-1})} w^{1/3})} \\ < \left(E_1^{(D)} \left(E_5^{(D)}(\delta) \right) \right)^{-2} \cdot 2^{O(d^{(3^{D+1}+D/3+D 3^{D-1})} (\text{polylog}(n))^{(3^{D+1}+D 3^{D-1})} w^{1/3})} \\ < \left(E_1^{(D)} \left(E_5^{(D)}(\delta) \right) \right)^{-2} \cdot 2^{O((d \text{polylog}(n))^{D 3^D} w^{1/3})}$$

Now note that

$$E_1(\delta) = 2^{\frac{\log(\delta)}{2h(n)} - 1} - 2^{\frac{\log(\delta)}{h(n)} - 1} = \frac{1}{2} \left(2^{\frac{\log(\delta)}{2h(n)}} - 2^{\frac{\log(\delta)}{h(n)}} \right) \geq \frac{\ln 2}{2} 2^{\frac{\log(\delta)}{h(n)}} \left(\frac{\log(\delta)}{2h(n)} - \frac{\log(\delta)}{h(n)} \right) = \\ -\frac{\log(\delta)}{4h(n)} 2^{\frac{\log(\delta)}{h(n)}} \cdot \ln 2$$

Hence, by monotonicity,

$$E_1(E_1(\delta)) = -\frac{\log(E_1(\delta))}{4h(n)} 2^{\frac{\log(E_1(\delta))}{h(n)}} \cdot \ln 2 \\ \geq -\frac{\log\left(-\frac{\log(\delta)}{4h(n)} 2^{\frac{\log(\delta)}{h(n)}} \cdot \ln 2\right)}{4h(n)} 2^{\frac{\log\left(-\frac{\log(\delta)}{4h(n)} 2^{\frac{\log(\delta)}{h(n)}} \cdot \ln 2\right)}{h(n)}} \cdot \ln 2 \\ = -\frac{\log\left(-\frac{\log(\delta)}{4h(n)}\right) + \log\left(2^{\frac{\log(\delta)}{h(n)}}\right) + \log(\ln 2)}{4h(n)} 2^{\frac{\log\left(-\frac{\log(\delta)}{4h(n)} 2^{\frac{\log(\delta)}{h(n)}} \cdot \ln 2\right)}{h(n)}} \cdot \ln 2$$

$$\begin{aligned}
&\geq -\frac{\log(\ln 2)}{4h(n)} 2^{\frac{\log\left(-\frac{\log(\delta)}{4h(n)} 2^{\frac{\log(\delta)}{h(n)} \cdot \ln 2}\right)}{h(n)}} \cdot \ln 2 \\
&= -\frac{\log(\ln 2)}{4h(n)} 2^{\frac{\log(E_1(\delta))}{h(n)}} \cdot \ln 2 \\
&\geq -\frac{\log(\ln 2)}{4h(n)} (E_1(\delta))^{\frac{1}{h(n)}} \cdot \ln 2 \\
&\geq -\frac{\log(\ln 2)}{4h(n)} E_1(\delta) \cdot \ln 2
\end{aligned}$$

And so for some constant a we get,

$$E_1(a\delta) \geq -\frac{\log(\ln 2)}{4h(n)} a\delta \cdot \ln 2$$

Therefore

$$\begin{aligned}
E_1^{(D)}(\delta) &\geq \left(\frac{-\log(\ln(2)) \ln(2)}{4h(n)}\right)^{D-1} E_1(\delta) \geq \left(\frac{-\log(\ln(2)) \ln(2)}{4h(n)}\right)^D \delta \\
\implies E_5(\delta) = E_3(E_2(E_1^{(D)}(\delta))) &\geq 2^{-10 \log(n) \log(\log(n)) - \Delta} \left(\frac{-\log(\ln(2)) \ln(2)}{4h(n)}\right)^D \delta \\
\implies E_5^{(D)}(\delta) &\geq 2^{D(-10 \log(n) \log(\log(n)) - \Delta)} \left(\frac{-\log(\ln(2)) \ln(2)}{4h(n)}\right)^{D^2} \delta \\
\implies (E_1^{(D)} \circ E_5^{(D)})(\delta) &\geq 2^{D(-10 \log(n) \log(\log(n)) - \Delta)} \left(\frac{-\log(\ln(2)) \ln(2)}{4h(n)}\right)^{D^2+D} \delta
\end{aligned}$$

Thus with $\Delta = \log(n)$ we get that

$$\begin{aligned}
\left((E_1^{(D)} \circ E_5^{(D)})(\delta)\right)^{-2} &\leq 2^{D(10 \log(n) \log(\log(n)) + \Delta)} \left(\frac{-\log(\ln(2)) \ln(2)}{4h(n)}\right)^{-2D^2-2D} \delta^{-2} \\
&= 2^{D \text{polylog}(n)} \left(\frac{-\log(\ln(2)) \ln(2)}{4h(n)}\right)^{-2D^2-2D} \delta^{-2}
\end{aligned}$$

Plugging this into our run-time bound we get

$$\begin{aligned}
T_1(\ell, D, d, w, \delta) &< 2^{D \text{polylog}(n)} \left(\frac{-\log(\ln(2)) \ln(2)}{4h(n)}\right)^{-2D^2-2D} \delta^{-2} \cdot 2^{O((d \text{polylog}(n))^{D^3} w^{1/3})} \\
&= \delta^{-2} \cdot 2^{O((d \text{polylog}(n))^{D^3} w^{1/3})} \quad \blacktriangleleft
\end{aligned}$$

► **Lemma 10.** $\mathcal{A}_{full}(S = (C, L, M, N), \mathcal{B}, \delta, D)$ returns an δ -additive error approximation of $|\langle 0_{ALL} | C | 0_{ALL} \rangle|^2$

Proof. Refer to Appendix A for proof. ◀

9:12 Approximating Output Probabilities of Shallow Quantum Circuits

■ **Algorithm 1** $\mathcal{A}_{full}(S = (C, L, M, N), \mathcal{B}, \delta, D)$: Quasi-Polynomial Time Additive Error Approximation for $|\langle 0_{ALL} | C | 0_{ALL} \rangle|^2$.

Input : Synthesis $S = (C, L, M, N)$ where C is a D -Dimensional Geometrically-Local depth- d circuit, \mathcal{B} a base case algorithm for 2D circuits, approximation error δ , dimension D

Output : An approximation of $|\langle 0_{ALL} | C | 0_{ALL} \rangle|^2$ to within additive error δ .

/* We begin by handling the case in which δ is so small that it trivializes our runtime, and the case in which δ is so large that it causes meaningless errors: */

1 **if** $\delta \leq 1/n^{\log^2(n)}$ **then**

2 **return** The value $|\langle 0_{ALL} | C | 0_{ALL} \rangle|^2$ computed with zero error by a “brute force” $2^{O(n)}$ -time algorithm.

3 **if** $\delta \geq 1/2$ **then return** $1/2$

4 **if** $D = 2$ **then**

5 **return** $\mathcal{B}(S, \delta)$

/* Here begins the non-trivial part of the algorithm: */

6 Let N be the register containing all of the qubits on which C acts. Since these qubits are arranged in a hyper-cubic lattice, the sides of the hyper-cube N must have length $n^{\frac{1}{D}}$. We will call the length of this side the “width” and will now describe how to “cut” the hyper-cube N , and the circuit C , perpendicular to this particular side.

7 Select $\frac{1}{10d}n^{\frac{1}{D}}$ light-cone separated slices K_i of $10d$ width in N , with at most $10d$ distance between adjacent slices. Let $h(n) = \log^7(n)$. Run Algorithm $\mathcal{A}_{full}(S, \mathcal{B}, 2^{\frac{\log(\delta)}{2h(n)}-1} - 2^{\frac{\log(\delta)}{h(n)}-1}, D - 1)$ to check if at least $\frac{1}{10d}n^{\frac{1}{D}} - h(n)$ of the slices obey:

8

$$|\text{tr}(\langle 0_{M_i} | C | 0_{ALL} \rangle \langle 0_{ALL} | C^\dagger | 0_{M_i} \rangle)| \geq 2^{\frac{\log(\delta)}{h(n)}}.$$

9 OR, there are fewer than $\frac{1}{10d}n^{\frac{1}{D}} - h(n)$ slices that obey:

10

$$|\text{tr}(\langle 0_{M_i} | C | 0_{ALL} \rangle \langle 0_{ALL} | C^\dagger | 0_{M_i} \rangle)| \geq 2^{\frac{\log(\delta)}{h(n)}}.$$

/* See the runtime analysis in the proof of Theorem 28 of [3] for a detailed explanation of how the aforementioned run of \mathcal{A}_{full} can efficiently distinguish between the above two cases (via Remark 6 in [3]). */

11 **if** Fewer than $\frac{1}{10d}n^{\frac{1}{D}} - h(n)$ of the slices obey Line 10 **then return** 0

12 **if** At least $\frac{1}{10d}n^{\frac{1}{D}} - h(n)$ of the slices obey Line 10 **then**

13 We will denote the set of these slices by K_{heavy} . Note that the maximum amount of width between any two adjacent slices in K_{heavy} is $10d \cdot h(n)$. Furthermore, the maximum amount of width collectively between Δ slices in K_{heavy} is $10d\Delta + 10d \cdot h(n)$. Now that the set K_{heavy} has been defined, we will use this fixed set in the recursive algorithm, Algorithm 2.

14 **return** $\mathcal{A}(S, \eta = \frac{\log(n)}{D \log(4/3)}, \Delta = \log(n), \epsilon = \delta 2^{-10 \log(n) \log(\log(n))}, h(n) = \log^7(n), K_{heavy}, D, \mathcal{B})$

■ **Algorithm 2** $\mathcal{A}(S, \eta, \Delta, \epsilon, h(n), K_{heavy}, D, \mathcal{B})$: Recursive Divide-and-Conquer Subroutine for Algorithm 1.

Input : D -dimensional Geometrically-Local, depth- d synthesis S , number of iterations η , number of cuts Δ , positive base-case error bound $\epsilon > 0$, a set of heavy slices K_{heavy} , dimension D , \mathcal{B} a base case algorithm for 2D circuits

Output : An approximation of the quantity $\langle 0_N | \phi_S | 0_N \rangle$ where ϕ_S is the un-normalized mixed state specified by the D -dimensional geometrically-local, depth- d synthesis S , and $|0_N\rangle$ is the 0 state on the entire N register of that synthesis.

- 1 Given the geometrically-local, depth- d synthesis $S = (\Gamma, L, M, N)$, let us ignore the registers L and M as they have already been measured or traced-out.
- 2 Let ℓ be the width of the N register of the synthesis S . Define the stopping width $w_0 \equiv 20d(\Delta + h(n) + 2)$.
- 3 **if** $\ell < w_0 = 20d(\Delta + h(n) + 2)$ **OR** $\eta < 1$ **then**
- 4 Compute the quantity $\langle 0_N | \phi_S | 0_N \rangle$ to within error ϵ .
- 5 **return** $\mathcal{A}_{full}(S, \mathcal{B}, \epsilon, D - 1)$
- 6 **else**
- 7 We will “slice” the D -Dimensional geometrically-local, depth- d synthesis S in Δ different locations, as follows:
- 8 Since N is D -Dimensional we define a region $Z \subset N$ to be the sub-hyper-cube of N which has width $10d(\Delta + h(n) + 2)$, and is centered at the halfway point of N width-wise (about the point $\ell/2$ of the way across N). Since the maximum amount of width collectively between Δ slices in K_{heavy} is $10d\Delta + 10d \cdot h(n)$ (see Algorithm 1), we are guaranteed that the region Z will contain at least Δ slices, $K_1, K_2, \dots, K_\Delta$, from K_{heavy} . For any two slices $K_i, K_j \in K_{heavy}$, let the un-normalized states $|\varphi_{L,i}\rangle, |\varphi_{i,j}\rangle, |\varphi_{j,R}\rangle$, and corresponding sub-syntheses $S_{L,i}, S_{i,j}, S_{j,R}$ be as defined in Definition 23 from [3], with $K = \log^3(n)$. We will use these to describe the result of our division step below.
- 9 For each $K_i \in K_{heavy}$ pre-compute the quantity $\kappa_{T,\epsilon}^i$, with $T = \log^3(n)$, and $\epsilon = \delta 2^{-10 \log(n) \log(\log(n))}$.
- 10 **return**

$$\sum_{i=1}^{\Delta} \frac{1}{(\kappa_{T,\epsilon}^i)^{4K+1}} \mathcal{A}(S_{L,i}, \eta - 1) \cdot \mathcal{A}(S_{i,R}, \eta - 1) \quad (3)$$

$$- \sum_{i=1}^{\Delta} \sum_{j=i+1}^{\Delta} \frac{1}{(\kappa_{T,\epsilon}^i \kappa_{T,\epsilon}^j)^{4K+1}} \mathcal{A}(S_{L,i}, \eta - 1) \cdot \mathcal{A}_{full}(S_{i,j}, \mathcal{B}, \epsilon, D - 1) \cdot \mathcal{A}(S_{j,R}, \eta - 1) \quad (4)$$

$$+ \sum_{i=1}^{\Delta} \sum_{j=i+2}^{\Delta} \frac{1}{(\kappa_{T,\epsilon}^i \kappa_{T,\epsilon}^j)^{4K+1}} \mathcal{A}(S_{L,i}, \eta - 1) \cdot \mathcal{A}(S_{j,R}, \eta - 1) \cdot \left[\sum_{\sigma \in \mathcal{P}(\{i+1, \dots, j-1\}) \setminus \emptyset} (-1)^{|\sigma|+1} \mathcal{A}_{full} \left(\left(\otimes_{k \in \sigma} \Pi_{F_k}^K \langle 0_{M_k} | \right) \phi_{i,j} \left(\otimes_{k \in \sigma} | 0_{M_k} \rangle \Pi_{F_k}^K \right), \mathcal{B}, \frac{\epsilon}{2^\Delta}, D - 1 \right) \right] \quad (5)$$

11

/* Note that for brevity it is implied that $\mathcal{A}(S, \eta) = \mathcal{A}(S, \eta, \Delta, \epsilon, h(n), K_{heavy}, D, \mathcal{B})$. */

References

- 1 Sergy Bravyi, David Gosset, and Ramis Movassagh. Classical algorithms for quantum mean values. QIP, 2020. [arXiv:1909.11485](#).
- 2 Michael J. Bremner, Richard Jozsa, and Dan J. Shepherd. Classical simulation of commuting quantum computations implies collapse of the polynomial hierarchy. *Proc. R. Soc. A.*, 467:459–472, 2010. [doi:10.1098/rspa.2010.0301](#).
- 3 Nolan J. Coble and Matthew Coudron. Quasi-polynomial time approximation of output probabilities of geometrically-local, shallow quantum circuits. In *62nd Annual Symposium on Foundations of Computer Science, FOCS 2021*, 2021.
- 4 András Gilyén, Yuan Su, Guang Hao Low, and Nathan Wiebe. Quantum singular value transformation and beyond: exponential improvements for quantum matrix arithmetics. STOC, 2019. [arXiv:1806.01838](#).
- 5 Yasuhiro Kondo, Ryuhei Mori, and Ramis Movassagh. Fine-grained analysis and improved robustness of quantum supremacy for random circuit sampling, 2021. [arXiv:2102.01960](#).
- 6 Ramis Movassagh. Quantum supremacy and random circuits. QIP, 2020. [arXiv:1909.06210](#).
- 7 Barbara M. Terhal and David P. DiVincenzo. Adaptive quantum computation, constant depth quantum circuits and arthur-merlin games. *Quantum Inf. Comput.*, 4(2):134–145, 2004. [doi:10.26421/QIC4.2-5](#).

A Proof of Lemma Statement

► **Lemma** (Restatement of Lemma 10). $\mathcal{A}_{full}(S = (C, L, M, N), \mathcal{B}, \delta, D)$ returns an δ -additive error approximation of $|\langle 0_{ALL} | C | 0_{ALL} \rangle|^2$

Proof. The error analysis of the error obtained by $\mathcal{A}_{full}(S, \mathcal{B}, \delta, D)$ can be broken into four cases according to the IF statements on Lines 1, 3, 11, and 12 of Algorithm 1. The first three cases can be easily shown to return the value in δ -additive error within the promised runtime as shown in page 16 and 20 of [3].

In the event that Line 12 is satisfied, Algorithm 1 returns the following quantity:

$$\mathcal{A}(S, \eta = \frac{\log(n)}{D \log(4/3)}, \Delta = \log(n), \epsilon = \delta 2^{-10 \log(n) \log(\log(n))}, h(n) = \log^7(n), K_{heavy}, D, \mathcal{B}),$$

which we know is an $f(S, \eta_D, \Delta, \epsilon, D)$ -additive error approximation of $|\langle 0_{ALL} | C | 0_{ALL} \rangle|^2$. Recall the definition of η_D defined in Equation 2. Since $\eta_D = \frac{\log(n)}{D \log(4/3)}$, by Equation 8, we know that:

$$\begin{aligned} f(S, \eta, \Delta, \epsilon) &\leq \eta_D (20\Delta^2)^{\eta_D} ((2e(n) + 2g(n))^\Delta + 3\Delta^2 \mathcal{E}_3(n, K, T, \epsilon, \Delta)) \\ &= \eta_D (20\Delta^2)^{\eta_D} 3\Delta^2 O(\mathcal{E}_3(n, K, T, \epsilon, \Delta)) \\ &= \eta_D (20\Delta^2)^{\eta_D} 3\Delta^2 O(2^\Delta (2e(n))^K + 2^\Delta K (e(n))^{2T} + \epsilon) + \epsilon \\ &= \frac{\log(n)}{D \log(4/3)} (20 \log^2(n))^{\frac{\log(n)}{D \log(4/3)}} 3 \log^2(n) O\left(2^{\log(n)} (2(1 - 2^{\frac{\log(\delta)}{\log^7(n)}})) \log^3(n)\right. \\ &\quad \left.+ 2^{\log(n)} \log^3(n) \left((1 - 2^{\frac{\log(\delta)}{\log^7(n)}})^2 \log^3(n) + \epsilon \right) + \delta 2^{-10 \log(n) \log(\log(n))} \right) \\ &\leq (\log(n))^{2 \log(n)} \cdot \text{poly}(n) \cdot \left((2(1 - 2^{\frac{\log(\delta)}{\log^7(n)}})) \log^3(n) + \epsilon + \delta 2^{-10 \log(n) \log(\log(n))} \right) \\ &\leq (\log(n))^{2 \log(n)} \cdot \text{poly}(n) \cdot \left(\left(O\left(\frac{1}{\log^4(n)}\right) \right)^{\log^3(n)} + 2 \cdot \delta 2^{-10 \log(n) \log(\log(n))} \right) \end{aligned}$$

$$\begin{aligned}
&\leq 2^{2 \log(n) \log(\log(n))} \cdot \text{poly}(n) \cdot \left(O \left(\frac{1}{\log^4(n)} \right) \right)^{\log^3(n)} + \delta 2^{-8 \log(n) \log(\log(n))} \\
&\leq o(1) \cdot \delta + o(1) \cdot \delta = o(1) \cdot \delta
\end{aligned} \tag{6}$$

where the first inequality follows from our result from the next subsection and the rest follows by calculation, noting that $E_3(n, K, T, \epsilon, \Delta) \geq (2e(n) + 2g(n))^\Delta$ for our specific choice of parameters (in particular $\Delta = \log(n)$). Note from [3] that $e(n) \leq (1 - 2^{\frac{\log(\delta)}{\log^7(n)}}) = O(1/\log^4(n))$ (since $\delta \geq n^{-\log^2(n)} = 2^{-\log(n)^3}$ as verified in Algorithm 1), $K = \log^3(n)$, $T = \log^3(n)$, and $\epsilon = \delta 2^{-10 \log(n) \log(\log(n))}$. The final inequality, which claims $2^{2 \log(n) \log(\log(n))} \cdot \text{poly}(n) \cdot \left(O \left(\frac{1}{\log^4(n)} \right) \right)^{\log^4(n)} = o(1) \cdot \delta$, again follows because $\delta \geq n^{-\log^2(n)}$ as verified in the driver algorithm, Algorithm 1.

As described on page 22 of [3], $\langle 0_{ALL} | \Psi_\theta \rangle \langle \Psi_\theta | 0_{ALL} \rangle$ is the quantity that we wish for Algorithm 2 to output. Refer to Definition 17 and Lemma 18 from [3] for the definition of $|\Psi_\theta\rangle$ and $|\Psi_\sigma\rangle$ for the subsequent analysis. Since Algorithm 2 depends on recursively calling itself, recall η_i from Equation 2 that defines the number of recursive calls for some dimension i . The error between the returned output of Algorithm 2, (defined on Line 10 of that algorithm) and the desired output quantity $\langle 0_{ALL} | \Psi_\theta \rangle \langle \Psi_\theta | 0_{ALL} \rangle$ is written below:

$$\begin{aligned}
f(S, \eta_D, \Delta, \epsilon, D) &\leq \left\| \langle 0_{ALL} | \Psi_\theta \rangle \langle \Psi_\theta | 0_{ALL} \rangle - \mathcal{A}(S, \eta_D, D, \epsilon) \right\| \\
&\leq \left\| \langle 0_{ALL} | \Psi_\theta \rangle \langle \Psi_\theta | 0_{ALL} \rangle - \sum_{\sigma \in \mathcal{P}([\Delta]) \setminus \emptyset} (-1)^{|\sigma|+1} \langle 0_{ALL} | \Psi_\sigma \rangle \langle \Psi_\sigma | 0_{ALL} \rangle \right\| \\
&+ \left\| \sum_{\sigma \in \mathcal{P}([\Delta]) \setminus \emptyset} (-1)^{|\sigma|+1} \langle 0_{ALL} | \Psi_\sigma \rangle \langle \Psi_\sigma | 0_{ALL} \rangle - \mathcal{A}(S, \eta_D, D, \epsilon) \right\| \\
&\leq (2e(n) + 2g(n))^\Delta + \left\| \sum_{\sigma \in \mathcal{P}([\Delta]) \setminus \emptyset} (-1)^{|\sigma|+1} \langle 0_{ALL} | \Psi_\sigma \rangle \langle \Psi_\sigma | 0_{ALL} \rangle - \mathcal{A}(S, \eta_D, D, \epsilon) \right\| \\
&= (2e(n) + 2g(n))^\Delta + \left\| \sum_{\sigma \in \mathcal{P}([\Delta]) \setminus \emptyset} (-1)^{|\sigma|+1} \langle 0_{ALL} | \Psi_\sigma \rangle \langle \Psi_\sigma | 0_{ALL} \rangle \right. \\
&- \left(\sum_{i=1}^{\Delta} \frac{1}{(\kappa_{T,\epsilon}^i)^{4K+1}} \mathcal{A}(S_{L,i}, \eta_D - 1, D, \epsilon) \cdot \mathcal{A}(S_{i,R}, \eta_D - 1, D, \epsilon) \right. \\
&- \sum_{i=1}^{\Delta} \sum_{j=i+1}^{\Delta} \frac{1}{(\kappa_{T,\epsilon}^i \kappa_{T,\epsilon}^j)^{4K+1}} \mathcal{A}(S_{L,i}, \eta_D - 1, D, \epsilon) \cdot \mathcal{A}_{full}(S_{i,j}, \mathcal{B}, D - 1, \epsilon) \cdot \mathcal{A}(S_{j,R}, \eta_D - 1, D, \epsilon) \\
&+ \sum_{i=1}^{\Delta} \sum_{j=i+2}^{\Delta} \frac{1}{(\kappa_{T,\epsilon}^i \kappa_{T,\epsilon}^j)^{4K+1}} \mathcal{A}(S_{L,i}, \eta_D - 1, D, \epsilon) \cdot \mathcal{A}(S_{j,R}, \eta_D - 1, D, \epsilon) \\
&\left. \cdot \left[\sum_{\sigma \in \mathcal{P}(\{i+1, \dots, j-1\}) \setminus \emptyset} (-1)^{|\sigma|+1} \mathcal{A}_{full} \left((\otimes_{k \in \sigma} \Pi_{F_k}^K \langle 0_{M_k} |) \phi_{i,j} (\otimes_{k \in \sigma} |0_{M_k}\rangle \Pi_{F_k}^K), \mathcal{B}, D - 1, E_3(\epsilon) \right) \right] \right\|
\end{aligned}$$

Grouping analogous terms and using triangle inequality gives:

$$\begin{aligned}
f(S, \eta_D, \Delta, \epsilon, D) &\leq (2e(n) + 2g(n))^\Delta \\
&+ \left\| \sum_{i=1}^{\Delta} \left(\langle 0_{ALL} | \Psi_{\{i\}} \rangle \langle \Psi_{\{i\}} | 0_{ALL} \rangle - \frac{1}{(\kappa_{T,\epsilon}^i)^{4K+1}} \mathcal{A}(S_{L,i}, \eta_D - 1, D, \epsilon) \cdot \mathcal{A}(S_{i,R}, \eta_D - 1, D, \epsilon) \right) \right. \\
&+ \sum_{i=1}^{\Delta} \sum_{j=i+1}^{\Delta} \left(\frac{1}{(\kappa_{T,\epsilon}^i \kappa_{T,\epsilon}^j)^{4K+1}} \mathcal{A}(S_{L,i}, \eta_D - 1, D, \epsilon) \cdot \mathcal{A}_{full}(S_{i,j}, \mathcal{B}, D - 1, \epsilon) \cdot \mathcal{A}(S_{j,R}, \eta_D - 1, D, \epsilon) \right. \\
&\quad \left. \left. - \langle 0_{ALL} | \Psi_{\{i,j\}} \rangle \langle \Psi_{\{i,j\}} | 0_{ALL} \rangle \right) \right\|
\end{aligned}$$

9:16 Approximating Output Probabilities of Shallow Quantum Circuits

$$\begin{aligned}
& - \sum_{i=1}^{\Delta} \sum_{j=i+2}^{\Delta} \sum_{\sigma \in \mathcal{P}(\{i+1, \dots, j-1\}) \setminus \emptyset} \left(\frac{1}{(\kappa_{T,\epsilon}^i \kappa_{T,\epsilon}^j)^{4K+1}} \mathcal{A}(S_{L,i}, \eta_D - 1, D, \epsilon) \right. \\
& \quad \cdot \mathcal{A}(S_{j,R}, \eta_D - 1, D, \epsilon) \cdot (-1)^{|\sigma|+1} \mathcal{A}_{full} \left(\left(\otimes_{k \in \sigma} \Pi_{F_k}^K \langle 0_{M_k} \right) \phi_{i,j} \left(\otimes_{k \in \sigma} |0_{M_k}\rangle \Pi_{F_k}^K \right), \mathcal{B}, D - 1, E_3(\epsilon) \right) \\
& \quad \left. - (-1)^{|\sigma|+1} \langle 0_{ALL} | \Psi_{\{i,j\} \cup \sigma} \rangle \langle \Psi_{\{i,j\} \cup \sigma} | 0_{ALL} \rangle \right) \Big\| \\
& \leq (2e(n) + 2g(n))^{\Delta} \\
& \quad + \sum_{i=1}^{\Delta} \left\| \left(\langle 0_{ALL} | \Psi_{\{i\}} \rangle \langle \Psi_{\{i\}} | 0_{ALL} \rangle - \frac{1}{(\kappa_{T,\epsilon}^i)^{4K+1}} \mathcal{A}(S_{L,i}, \eta_D - 1, D, \epsilon) \cdot \mathcal{A}(S_{i,R}, \eta_D - 1, D, \epsilon) \right) \right\| \\
& \quad + \sum_{i=1}^{\Delta} \sum_{j=i+1}^{\Delta} \left\| \left(\frac{1}{(\kappa_{T,\epsilon}^i \kappa_{T,\epsilon}^j)^{4K+1}} \mathcal{A}(S_{L,i}, \eta_D - 1, D, \epsilon) \cdot \mathcal{A}_{full}(S_{i,j}, \mathcal{B}, D - 1, E_3(\epsilon)) \cdot \mathcal{A}(S_{j,R}, \eta_D - 1, D, \epsilon) \right. \right. \\
& \quad \left. \left. - \langle 0_{ALL} | \Psi_{\{i,j\}} \rangle \langle \Psi_{\{i,j\}} | 0_{ALL} \rangle \right) \right\| \\
& \quad - \sum_{i=1}^{\Delta} \sum_{j=i+2}^{\Delta} \left\| \sum_{\sigma \in \mathcal{P}(\{i+1, \dots, j-1\}) \setminus \emptyset} (-1)^{|\sigma|+1} \left(\frac{1}{(\kappa_{T,\epsilon}^i \kappa_{T,\epsilon}^j)^{4K+1}} \mathcal{A}(S_{L,i}, \eta_D - 1, D, \epsilon) \cdot \mathcal{A}(S_{j,R}, \eta_D - 1, D, \epsilon) \right. \right. \\
& \quad \left. \left. \cdot \mathcal{A}_{full} \left(\left(\otimes_{k \in \sigma} \Pi_{F_k}^K \langle 0_{M_k} \right) \phi_{i,j} \left(\otimes_{k \in \sigma} |0_{M_k}\rangle \Pi_{F_k}^K \right), \mathcal{B}, D - 1, E_3(\epsilon) \right) \right. \right. \\
& \quad \left. \left. - \langle 0_{ALL} | \Psi_{\{i,j\} \cup \sigma} \rangle \langle \Psi_{\{i,j\} \cup \sigma} | 0_{ALL} \rangle \right) \right\| \tag{7}
\end{aligned}$$

We will now use Lemma 11, 12, and 13 that are adapted versions of Lemma 29, 30, and 31 from [3] to bound the last three terms of the above inequality. Because their bounds are independent of dimensions, the proofs for the three lemmas will be similar to the proofs in [3].

$$\begin{aligned}
f(S, \eta_D, \Delta, \epsilon, D) & \leq (2e(n) + 2g(n))^{\Delta} + \Delta (\mathcal{E}_1(n, K, T, \epsilon) + 2f(S, \eta_D - 1, \Delta, \epsilon, D)) \\
& \quad + \Delta^2 (\mathcal{E}_2(n, K, T, \epsilon) + 2f(S, \eta_D - 1, \Delta, \epsilon, D)) \\
& \quad + \Delta^2 (\mathcal{E}_3(n, K, T, \epsilon, \Delta) + 16f(S, \eta_D - 1, \Delta, \epsilon, D)) \\
& \leq (2e(n) + 2g(n))^{\Delta} + 3\Delta^2 \mathcal{E}_3(n, K, T, \epsilon, \Delta) + 20\Delta^2 f(S, \eta_D - 1, \Delta, \epsilon, D) \\
& = (2e(n) + 2g(n))^{\Delta} + 3\Delta^2 \mathcal{E}_3(n, K, T, \epsilon, \Delta) \\
& \quad + 20\Delta^2 \left[(2e(n) + 2g(n))^{\Delta} + 3\Delta^2 \mathcal{E}_3(n, K, T, \epsilon, \Delta) + 20\Delta^2 f(S, \eta_D - 2, \Delta, \epsilon, D) \right] \\
& = \sum_{i=0}^{\eta_D - 1} \left[(20\Delta^2)^i \left((2e(n) + 2g(n))^{\Delta} + 3\Delta^2 \mathcal{E}_3(n, K, T, \epsilon, \Delta) \right) \right] + (20\Delta^2)^{\eta_D} f(S, 0, \Delta, \epsilon, D) \\
& \leq \eta_D (20\Delta^2)^{\eta_D} \left((2e(n) + 2g(n))^{\Delta} + 3\Delta^2 \mathcal{E}_3(n, K, T, \epsilon, \Delta) \right) + (20\Delta^2)^{\eta_D} \epsilon \\
& \leq \eta_D (20\Delta^2)^{\eta_D} (\epsilon + (2e(n) + 2g(n))^{\Delta} + 3\Delta^2 \mathcal{E}_3(n, K, T, \epsilon, \Delta)) \\
& \leq \eta_D (20\Delta^2)^{\eta_D} ((2e(n) + 2g(n))^{\Delta} + 3\Delta^2 \mathcal{E}_3(n, K, T, \epsilon, \Delta)) \tag{8}
\end{aligned}$$

where the above inequalities follow because $\mathcal{E}_3(n, K, T, \epsilon, \Delta) \geq \mathcal{E}_2(n, K, T, \epsilon) \geq \mathcal{E}_1(n, K, T, \epsilon)$ and $f(S, 0, \Delta, \epsilon, \cdot) \leq \epsilon \leq \mathcal{E}_3(n, K, T, \epsilon, \Delta)$ \blacktriangleleft

► **Lemma 11.**

$$\left\| \left(\frac{1}{(\kappa_{T,\epsilon}^i)^{4K+1}} \mathcal{A}(S_{L,i}, \eta_D - 1, D, \epsilon) \cdot \mathcal{A}(S_{i,R}, \eta_D - 1, D, \epsilon) - \langle 0_{ALL} | \Psi_{\{i\}} \rangle \langle \Psi_{\{i\}} | 0_{ALL} \rangle \right) \right\|$$

$$\leq \mathcal{E}_1(n, K, T, \epsilon) + 2f(S, \eta_D - 1, \Delta, \epsilon, D),$$

where $\mathcal{E}_1(n, K, T, \epsilon) \equiv 10K(e(n)^{2T} + 6g(n) + \epsilon)$.

► **Lemma 12.**

$$\left\| \left(\frac{1}{(\kappa_{T,\epsilon}^i \kappa_{T,\epsilon}^j)^{4K+1}} \mathcal{A}(S_{L,i}, \eta_D - 1, D, \epsilon) \cdot \mathcal{A}_{full}(S_{i,j}, \mathcal{B}, D - 1, \epsilon) \cdot \mathcal{A}(S_{j,R}, \eta_D - 1, D, \epsilon) \right. \right.$$

$$\left. - \langle 0_{ALL} | \Psi_{\{i,j\}} \rangle \langle \Psi_{\{i,j\}} | 0_{ALL} \rangle \right\|$$

$$\leq \mathcal{E}_2(n, K, T, \epsilon) + 2f(S, \eta_D - 1, \Delta, \epsilon, D), \quad (9)$$

where $\mathcal{E}_2(n, K, T, \epsilon) \equiv 10K(e(n)^{2T} + 6g(n) + \epsilon) + \epsilon$

► **Lemma 13.**

$$\left\| \sum_{\sigma \in \mathcal{P}(\{i+1, \dots, j-1\}) \setminus \emptyset} (-1)^{|\sigma|+1} \left(\frac{1}{(\kappa_{T,\epsilon}^i \kappa_{T,\epsilon}^j)^{4K+1}} \mathcal{A}(S_{L,i}, \eta_D - 1, D, \epsilon) \cdot \mathcal{A}(S_{j,R}, \eta_D - 1, D, \epsilon) \right. \right.$$

$$\left. \cdot \mathcal{A}_{full} \left(\left(\otimes_{k \in \sigma} \Pi_{F_k}^K |0_{M_k}\rangle \right) \phi_{i,j} \left(\otimes_{k \in \sigma} |0_{M_k}\rangle \Pi_{F_k}^K \right), \mathcal{B}, D - 1, E_3(\epsilon) \right) \right.$$

$$\left. - \langle 0_{ALL} | \Psi_{\{i,j\} \cup \sigma} \rangle \langle \Psi_{\{i,j\} \cup \sigma} | 0_{ALL} \rangle \right\| \quad (10)$$

$$\leq \mathcal{E}_3(n, K, T, \epsilon, \Delta) + 16f(S, \eta_D - 1, \Delta, \epsilon, D),$$

where $\mathcal{E}_3(n, K, T, \epsilon, \Delta) \equiv O(2^\Delta(6g(n)) + 2^\Delta K(e(n)^{2T} + \epsilon) + \epsilon)$

Memory Compression with Quantum Random-Access Gates

Harry Buhrman ✉

QuSoft, CWI Amsterdam, The Netherlands
University of Amsterdam, The Netherlands

Bruno Loff ✉

University of Porto, Portugal
INESC-Tec, Porto, Portugal

Subhasree Patro ✉

QuSoft, CWI Amsterdam, The Netherlands
University of Amsterdam, The Netherlands

Florian Speelman ✉

QuSoft, CWI Amsterdam, The Netherlands
University of Amsterdam, The Netherlands

Abstract

In the classical RAM, we have the following useful property. If we have an algorithm that uses M memory cells throughout its execution, and in addition is sparse, in the sense that, at any point in time, only m out of M cells will be non-zero, then we may “compress” it into another algorithm which uses only $m \log M$ memory and runs in almost the same time. We may do so by simulating the memory using either a hash table, or a self-balancing tree.

We show an analogous result for quantum algorithms equipped with quantum random-access gates. If we have a quantum algorithm that runs in time T and uses M qubits, such that the state of the memory, at any time step, is supported on computational-basis vectors of Hamming weight at most m , then it can be simulated by another algorithm which uses only $O(m \log M)$ memory, and runs in time $\tilde{O}(T)$.

We show how this theorem can be used, in a black-box way, to simplify the presentation in several papers. Broadly speaking, when there exists a need for a space-efficient history-independent quantum data-structure, it is often possible to construct a space-inefficient, yet sparse, quantum data structure, and then appeal to our main theorem. This results in simpler and shorter arguments.

2012 ACM Subject Classification Theory of computation → Quantum complexity theory; Theory of computation → Quantum complexity theory

Keywords and phrases complexity theory, data structures, algorithms, quantum walk

Digital Object Identifier 10.4230/LIPIcs.TQC.2022.10

Related Version *ArXiv Version*: <https://arxiv.org/abs/2203.05599>

Funding This work was supported by the Dutch Ministry of Economic Affairs and Climate Policy (EZK), as part of the Quantum Delta NL programme.

Harry Buhrman: Harry Buhrman is supported by NWO Gravitation grants NETWORKS and QSC, and EU grant QuantAlgo.

Bruno Loff: Bruno Loff’s research is supported by National Funds through the Portuguese funding agency, FCT – Fundação para a Ciência e a Tecnologia, within project LA/P/0063/2020.

Subhasree Patro: Subhasree Patro is supported by the Robert Bosch Stiftung and additionally supported by NWO Gravitation grants NETWORKS and QSC, and EU grant QuantAlgo.

Florian Speelman: Florian Speelman is supported by NWO Gravitation grants NETWORKS and QSC, and EU grant QuantAlgo.

Acknowledgements We would like to thank Robin Kothari and Ryan O’Donnell for helpful discussions.



© Harry Buhrman, Bruno Loff, Subhasree Patro, and Florian Speelman;
licensed under Creative Commons License CC-BY 4.0

17th Conference on the Theory of Quantum Computation, Communication and Cryptography (TQC 2022).

Editors: François Le Gall and Tomoyuki Morimae; Article No. 10; pp. 10:1–10:19

Leibniz International Proceedings in Informatics



LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

1 Introduction

This paper arose out of the authors' recent work on quantum fine-grained complexity [4], where we had to make use of a quantum walk, similar to how Ambainis uses a quantum walk in his algorithm for element distinctness [2]. An essential aspect of these algorithms is the use of a history-independent data-structure. In the context of our paper, we needed three slightly different data structures of this type, and on each of these occasions we saw a similar scenario. If we were only concerned with the time complexity of our algorithm, and were OK with a polynomial increase in the space complexity (the number of qubits used by the algorithm), then there was a very simple data structure that would serve our purpose. If, however, we wanted the algorithm to be space-efficient, as well, then we needed to resort to more complicated data structures.

And we made the following further observation: the simple, yet space-inefficient, data structures were actually *sparse*, in the sense that although M qubits were being used, all the amplitude was always concentrated on computational-basis vectors of Hamming weight $\leq m \ll M$. The analogous classical scenario is an algorithm that uses M memory registers, but at any time step all but m of these registers are set to 0. In the classical case, we know how to convert any such an *m-sparse* algorithm into an algorithm that uses $O(m \log M)$ memory, by using, e.g., a hash table. We wondered whether the same thing could be said of quantum algorithms. This turned out to be possible, and the main purpose of this paper is to explain how it can be done. We will take an arbitrary *sparse* quantum algorithm, and *compress* it into a quantum algorithm that uses little space.

Our main theorem is as follows (informally stated):

► **Theorem 1.** *Any m -sparse quantum algorithm using time T and M qubits can be simulated with ε additional error by a quantum algorithm running in time $O(T \cdot \log(\frac{T}{\varepsilon}) \cdot \log(M))$, using $O(m \log M)$ qubits.*

We will prove this result using quantum radix trees in Section 3. The result can also be proven, with slightly worse parameters, using hash tables, but we will not do so here. The sparse algorithm is allowed to use quantum random-access gates (described in the preliminaries Section 2), and the *compressed* simulation requires such gates, even if the original algorithm does not.

The $\log M$ factor in the time bound can be removed if we assume that certain operations on $O(\log M)$ bits can be done at $O(1)$ cost. This includes only simple operations such as comparison, addition, bitwise XOR, or swapping of two blocks of $O(\log M)$ adjacent qubits.¹ All these operations can be done at $O(1)$ cost in the usual classical Random-Access Machines.

The techniques used to prove our main theorem are not new: quantum radix-trees first appeared in a paper by Bernstein, Jeffery, Lange and Meurer [3] (see also Jeffery's PhD thesis [5]). One contribution of our paper is to present BJLM technique in full, as in currently available presentations of the technique, several crucial aspects of the implementation are missing or buggy².

But our main contribution is to use these techniques at the right level of abstraction. Theorem 1 is very general, and can effectively be used as a black box. One would think that Theorem 1, being such a basic and fundamental statement about quantum computers, and being provable essentially by known techniques, would already be widely known. But this

¹ The qubits in each block are adjacent, but the two swapped blocks can be far apart from each other.

² For example, some operations are defined which are not unitary. Or, there is no mention of error in the algorithms, but they actually cannot be implemented in an error-free way using a reasonable number of gates from any standard gate set.

appears not to be the case, as papers written as recently as a year ago could be significantly simplified by appealing to such a theorem. Indeed, we believe that the use of Theorem 1 will save researchers a lot of work in the future, and this is our main motivation for writing this paper.

To illustrate this point, in Section 4 we will overview three papers [2, 1, 4] that make use of a quantum walk together with a history-independent data structure. These papers all use complicated but space-efficient data structures. As it turns out, we can replace these complicated data structures with very simple tree-like data structures. These new, simple data structures are memory inefficient but sparse, so we may then appeal to Theorem 1 to get similar upper bounds. The proofs become shorter: we estimate each of these papers could be cut in size by 4 to 12 pages. And furthermore, using simpler (memory inefficient but sparse) data-structures allows for a certain *separation of concerns*: when one tries to describe a space-efficient algorithm, there are several bothersome details that one needs to keep track of, and they obscure the presentation of the algorithm. By using simpler data structures, these bothersome details are disappear from the proofs, and are entrusted to Theorem 1.

2 Definitions

We let $[n] = \{1, \dots, n\}$, and let $\binom{[n]}{\leq m}$ be the set of subsets of $[n]$ of size at most m .

We let $\mathcal{H}(N)$ denote the complex Hilbert space of dimension N , and we let $\mathcal{U}(N)$ denote the space of unitary linear operators from $\mathcal{H}(N)$ to itself (i.e. the unitary group). We let \mathcal{B} denote a set of universal quantum gates, which we will fix to containing the $\text{CNOT} \in \mathcal{U}(4)$ and all single-qubit gates, but which we could have been chosen from among any of the standard possibilities.

Of particular importance to this paper will be the set $\mathcal{Q} = \mathcal{B} \cup \{\text{RAG}_n \mid n \text{ a power of } 2\}$ which contains our universal set together with the *random-access gates*, so that $\text{RAG}_n \in \mathcal{U}(n2^{1+n})$ is defined on the computational basis by:

$$\begin{aligned} \text{RAG}_n |i, b, x_0, \dots, x_{n-1}\rangle &= |i, x_i, x_0, \dots, x_{i-1}, b, x_{i+1}, \dots, x_{n-1}\rangle \\ &\quad \forall i \in [n], b, x_0, \dots, x_{n-1} \in \{0, 1\} \end{aligned}$$

We now give a formal definition of what it means to solve a Boolean relation $F \subseteq \{0, 1\}^n \times \{0, 1\}^m$ using a quantum circuit. This includes the special case when F is a function.

A *quantum circuit* over a gate set \mathcal{G} (such as \mathcal{B} or \mathcal{Q}) is a tuple $C = (n, T, S, C_1, \dots, C_T)$, where $T \geq 0$, $n, S \geq 1$ are natural numbers, and the C_t give us a sequence of instructions. Each instruction C_t comes from a set $\mathcal{I}_{\mathcal{G}}(S)$ of possible instructions, defined below. The number n is the input length, the number T is the *time complexity*, and S is the *space complexity*, also called the *number of wires* or the *number of ancillary qubits* of the circuit. Given an input $x \in \{0, 1\}^n$, at each step $t \in \{0, \dots, T\}$ of computation, the circuit produces an S -qubit state $|\psi_t(x)\rangle \in \mathcal{H}(2^S)$, starting with $|\psi_0(x)\rangle = |0\rangle^{\otimes S}$, and then applying each instruction C_t , as we will now describe.

For each possible q -qubit gate $G \in \mathcal{G} \cap \mathcal{U}(2^q)$, and each possible ordered choice $I = (i_1, \dots, i_q) \in [S]^q$ of distinct q among S qubits, we have an instruction $\text{APPLY}_{G,I} \in \mathcal{I}_{\mathcal{G}}(S)$ which applies gate G to the qubits indexed by I , in the prescribed order. The effect of executing the instruction $\text{APPLY}_{G,I}$ on $|\psi\rangle \in \mathcal{H}(2^S)$ is to apply G on the qubits indexed by I , tensored with identity on the remaining $S - q$ qubits. I.e., $\text{APPLY}_{G,I} \in \mathcal{U}(2^S)$ corresponds to the unitary transformation defined on each basis state by:

$$\text{APPLY}_{G,I} \cdot |y_I\rangle \otimes |y_J\rangle = (G|y_I\rangle) \otimes |y_J\rangle,$$

where $J = [S] \setminus I$.

10:4 Memory Compression with Quantum Random-Access Gates

Furthermore, for each possible ordered choice $I = (i_1, \dots, i_{\lceil \log n \rceil}) \in [S]^{\lceil \log n \rceil}$ of distinct $\lceil \log n \rceil$ among S qubits, and each $i \in [S] \setminus I$, we have an instruction $\text{READ}_{I,i} \in \mathcal{I}_{\mathcal{G}}(S)$, which applies the query oracle on the qubits indexed by I and i . I.e., given an input $x \in \{0, 1\}^n$, the instruction $\text{READ}_{I,i} \in \mathcal{U}(2^S)$ applies the unitary transformation defined on each basis state by:

$$\text{READ}_{I,i} \cdot |y_I\rangle \otimes |y_i\rangle \otimes |y_J\rangle = |y_I\rangle \otimes |y_i \oplus x_{y_I}\rangle \otimes |y_J\rangle,$$

where $J = [S] \setminus (I \cup \{i\})$.

Hence if we have a sequence C_1, \dots, C_T of instructions and an input x , we may obtain the state of the memory at time step t , on input x , by $|\psi_0(x)\rangle = |0\rangle^{\otimes S}$ and $|\psi_{t+1}(x)\rangle = C_{t+1}|\psi_t(x)\rangle$.

We say that a quantum circuit $C = (n, T, S, C_1, \dots, C_T)$ *computes* or *solves a relation* $F \subseteq \{0, 1\}^n \times \{0, 1\}^m$ *with error* ε if C is such that, for every input $x \in \{0, 1\}^n$, if we measure the first m qubits of $|\psi_T(x)\rangle$ in the computational basis, we obtain, with probability $\geq 1 - \varepsilon$, a string $z \in \{0, 1\}^m$ such that $(x, z) \in F$.

2.1 Quantum Random-Access Machine (QRAM)

Generally speaking, a quantum circuit is allowed to apply any of the basic operations to any of its qubits. In the definition given above, a quantum random-access gate can specify any permuted subset of the qubits to serve as its inputs. This allows for unusual circuit architectures, which are undesirable.

One may then define a more restricted class of circuits, as follows. We think of the qubits as divided into two parts: work qubits and memory qubits. We have M memory qubits and $W = O(\log M)$ work qubits, for a total space complexity $S = W + M$. We restrict the circuit so that any unitary gate $G \in \mathcal{B}$, or read instruction, must be applied to work qubits only. And, finally, any random-access gate must be applied in such a way that the addressing qubits (i) and the swap qubit (b) are always the first $\log M + 1$ work qubits, and the addressed qubits (x_0, \dots, x_{M-1}) are exactly the memory qubits, and are always addressed in the same, fixed order, so one can speak of *the first memory qubit, the second memory qubit, etc.* We may then think of a computation as alternating between doing some computation on the work registers, then swapping some qubits between work and memory registers, then doing some more computation on the work registers, and so forth. The final computational-basis measurement is also restricted to measuring a subset of the work qubits.

Under these restrictions, a circuit of time complexity T may be encoded using $O(T \log S)$ bits, whereas in general one might need $\Omega(TS)$ qubits in order to specify how the wires of the circuit connect to the random-access gates.

We will then use the term a *quantum random-access machine algorithm*, or *QRAM algorithm*, for a family of circuits that operate under these restrictions.³

2.2 Sparse QRAM algorithms

In classical algorithms, we may have an algorithm which uses M memory registers, but such that, at any given time, only m out of these M registers are non-zero. In this case we could call such an algorithm *m-sparse*. The following definition is the quantum analogue of this.

³ Such a computational model has been referred to by several names in the past. For instance, the term *QRAM* appears in several publications, starting with [6], and *QAQM* has also been used [7].

► **Definition 2.** Let $\mathcal{C} = (n, T, W, M, C_1, \dots, C_T)$ be a QRAM algorithm using time T , W work qubits, and M memory qubits. Then, we say that \mathcal{C} is m -sparse, for some $m \leq M$, if at every time-step $t \in \{0, \dots, T\}$ of the algorithm, the state of the memory qubits is supported on computational basis vectors of Hamming weight $\leq m$. I.e., we always have

$$|\psi_t\rangle \in \text{span} \left(|u\rangle|v\rangle \mid u \in \{0, 1\}^W, v \in \binom{[M]}{\leq m} \right)$$

In other words, if $|\psi_t\rangle$ is written in the computational basis:

$$|\psi_t\rangle = \sum_{u \in \{0, 1\}^W} \sum_{v \in \{0, 1\}^M} \alpha_{u,v}^{(t)} \cdot \underbrace{|u\rangle}_{\text{Work qubits}} \otimes \underbrace{|v\rangle}_{\text{Memory qubits}},$$

then $\alpha_{u,v}^{(t)} = 0$ whenever $|v| > m$.

2.3 Time complexity of simple operations (the constant γ)

Throughout the paper we will often describe algorithms that use certain simple operations over a logarithmic number of bits. These may include comparison, addition, bitwise XOR, swapping, and others. In a classical random-access machine, all of these operations can be done in $O(1)$ time, as in such machines it is usually considered that every memory position is a register that can hold $O(\log n)$ bits, and such simple operations are taken to be machine instructions.

We do not necessarily wish to make such an assumption for quantum algorithms, since we do not really know what a quantum computer will look like, just yet. So we will broadly postulate the existence of a quantity γ , which is an upper-bound on the time complexity of doing such simple operations. We then express our time upper-bounds with γ as a parameter. Depending on the precise architecture of the quantum computer, one may think of γ as being $O(1)$, or $O(\log n)$. In all our bounds, the simple operations that we will make use of can always be implemented using $O(\log M)$ elementary gates.

2.4 Controlled unitaries

Sometimes we will explain how to implement a certain unitary, and we wish to have a version of the same unitary which can be activated or deactivated depending on the state of an additional control bit. We will make free use of the following lemma, which we state without proof.

► **Lemma 3.** If a unitary U can be implemented using T gates from \mathcal{Q} , then the unitary

$$|b\rangle|x\rangle \mapsto \begin{cases} |b\rangle(U|x\rangle) & \text{if } b = 1 \\ |b\rangle|x\rangle & \text{if } b = 0 \end{cases}$$

can be implemented (without error) using $O(T)$ gates from \mathcal{Q} .

3 Compressing sparse algorithms using quantum radix trees

Let $\mathcal{C} = (n, T, W, M, C_1, \dots, C_T)$ be the circuit of an m -sparse QRAM algorithm computing a relation F with error ε and let the state of the algorithm at every time-step t , when written in the computational basis, be

$$|\psi_t\rangle = \sum_{u \in \{0, 1\}^W} \sum_{v \in \binom{[M]}{\leq m}} \alpha_{u,v}^{(t)} \underbrace{|u\rangle}_{\text{Work qubits}} \otimes \underbrace{|v\rangle}_{\text{Memory qubits}}. \quad (1)$$

10:6 Memory Compression with Quantum Random-Access Gates

Using the description of \mathcal{C} and the assumption that this algorithm is m -sparse we will now construct another QRAM algorithm \mathcal{C}' that uses much less space ($O(m \log M)$ qubits) and computes F with almost same error probability with only $O(\log M \log T)$ factor worsening in the run time.

Main observation

As the state of the memory qubits in $|\psi_t\rangle$ for any t is only supported on computational basis vectors of Hamming weight at most m , one immediate way to improve on the space complexity is to succinctly represent the state of the sparsely used memory qubits. The challenge, however, is that every instruction C_i in \mathcal{C} might not have an *easy* analogous implementation in the succinct representation. So we will first present a succinct representation and then show that, for every instruction C_i in the original circuit \mathcal{C} , there is an analogous instruction or a series of instructions that evolve the state of the succinct representation in the same way as the original state evolves due to the application of C_i .

A succinct representation

Let $v \in \{0, 1\}^M$ be a vector with $|v| \leq m$ (with $|v\rangle$ being the corresponding quantum state that uses M qubits). Whenever m is significantly smaller than M (i.e., $m \log M < M$) we can instead represent the vector v using the list of indices $\{i\}$ such that $v[i] = 1$. Such a representation will use much fewer (qu)bits. Let S_v denote the set of indices i such that $v[i] = 1$. We will then devise a quantum state $|S_v\rangle$, that represents the set S_v using a quantum data structure. This representation will be unique, meaning that for every sparse computational-basis state $|v\rangle$ there will be a unique corresponding quantum state $|S_v\rangle$, and $|S_v\rangle$ will use much fewer qubits. Then for every time-step t , the quantum state $|\psi_t\rangle$ from Equation (1) has a corresponding *succinctly represented* quantum state $|\phi_t\rangle$ such that

$$|\phi_t\rangle = \sum_{u \in \{0,1\}^w} \sum_{v \in \binom{[M]}{\leq m}} \alpha_{u,v}^{(t)} |u\rangle \otimes |S_v\rangle. \quad (2)$$

By using such a succinct representation, we will be able to simulate the algorithm \mathcal{C} with $O(m \log M)$ qubits, with an $O(\gamma \log \frac{T}{\delta})$ additional factor overhead in time and an additional δ probability of error.

To obtain the desired succinct representation $|S_v\rangle$, we use the quantum radix trees appearing in an algorithm for the subset-sum problem by Bernstein, Jeffery, Lange, and Meurer [3] (see also [5]). Several crucial aspects of the implementation were missing or buggy, and required some amount of work to complete and fix. The resulting effort revealed, in particular, that the data-structure is unlikely to be implementable efficiently without error (as it relies on a particular gate which cannot be implemented in an error-free way using the usual basic gates). So we here include all the required details.

3.1 Radix Tree

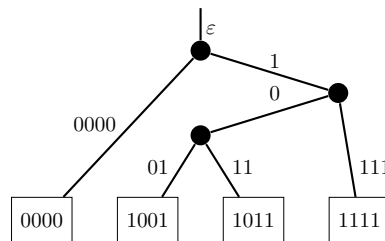
A quantum radix tree is a quantum data structure inspired by the classical radix tree whose definition is as follows.

► **Definition 4.** A radix tree is a rooted binary tree, where the edges are labeled by non-empty binary strings, and the concatenation of the labels of the edges along any root-to-leaf path results in a string of the same length ℓ (independent of the chosen root-to-leaf path). The value ℓ is called the word length of the tree.

There is a bijective correspondence between radix trees R of word length ℓ and subsets $S \subseteq \{0, 1\}^\ell$. Given R , we may obtain S as follows. Each root-to-leaf path of R gives us an element $x \in S$, so that x is the concatenation of all the edge labels along the path.

If R corresponds to S , we say that R stores, or represents S , and write $R(S)$ for the radix tree representing S , i.e., for the inverse map of what was just described (see below).

An example of a radix tree appears in Figure 1.



■ **Figure 1** A radix tree storing the set $\{0000, 1001, 1011, 1111\}$.

Given a set $S \subseteq \{0, 1\}^\ell$, we obtain $R(S)$ recursively as follows: The empty set corresponds to the tree having only the root and no other nodes. We first find the longest common prefix $p \in \{0, 1\}^{\leq \ell}$ of S . If $|p| > 0$, then we have a single child under the root, with a p -labeled edge going into it, which itself serves as the root to $R(S')$, where S' is the set of suffixes (after p) of S . If $|p| = 0$, then the root will have two children. Let $S = S_0 \cup S_1$, where S_0 and S_1 are sets of strings starting with 0 and 1, respectively, in S . The edges to the left and right children will be labeled by p_0 and p_1 , respectively, where $p_0 \in \{0, 1\}^{\leq \ell}$ is the longest common prefix of S_0 and $p_1 \in \{0, 1\}^{\leq \ell}$ is the longest common prefix of S_1 . The left child serves as a root to $R(S'_0)$, where S'_0 is the set of suffixes (after p_0) of S_0 . Analogously, the right child serves as a root to $R(S'_1)$, where S'_1 is the set of suffixes (after p_1) of S_1 .

Basic operations on radix trees

The allowed basic operations on a radix tree are insertion and removal of an element. Classically, an attempt at inserting an element already in S will result in the identity operation. Quantumly, we will instead allow for *toggling* an element in/out of S .

Representing a radix tree in memory

We now consider how one might represent a radix tree in memory. For this purpose, suppose we wish to represent a radix tree $R(S)$ for some set $S \subseteq \{0, 1\}^\ell$ of size $|S| \leq m$. Let us assume without loss of generality that m is a power of 2, and suppose we have at our disposal an array of $2m$ memory blocks.

Each memory block may be used to store a node of the radix tree. If we have a node in the tree, the contents of its corresponding memory block will represent a tuple (z, p_1, p_2, p_3) . The value $z \in \{0, 1\}^{\leq \ell}$ stores the label in the edge from the node's parent, the values $p_1, p_2, p_3 \in \{0, 1, \dots, 2m\}$ are pointers to the (block storing the) parent, left child, and right child, respectively, or 0 if such an edge is absent.

10:8 Memory Compression with Quantum Random-Access Gates

It follows that each memory block is $O(\ell + \log m)$ bits long. In this way, we will represent $R(S)$ by a binary string of length $O(m(\ell + \log m))$. The root node is stored in the first block, empty blocks will be set to 0, and the only thing that needs to be specified is the *memory layout*, namely, in which block does each node get stored. For this purpose, let $\tau : R(S) \rightarrow [2m]$ be an injective function, mapping the nodes of $R(S)$ to the $[2m]$ memory blocks, so that $\tau(\text{root}) = 1$. For any $S \subseteq \{0, 1\}^\ell$ of size $|S| \leq m$, we then let

$$R_\tau(S) \in \{0, 1\}^{O(m(\ell + \log m))}$$

denote the binary string obtained by encoding $R(S)$ as just described.

BJLM's quantum radix tree

We see now that although there is a unique radix tree $R(S)$ for each S , there is no obvious way of making sure that the representation of $R(S)$ in memory is also unique. However, this bijective correspondence between S and its memory representation is a requirement for quantum algorithms to use interference. The idea of Bernstein et al [3], then, is to represent S using a superposition of *all possible layouts*. I.e., S is to be uniquely represented by the (properly normalized) quantum state:

$$\sum_{\tau} |R_\tau(S)\rangle.$$

The trick, then, is to ensure that this representation can be efficiently queried and updated. In their discussion of how this might be done, the BJLM paper [3] presents the broad idea but does not work out the details, whereas Jeffery's thesis [5] glosses over several details and includes numerous bugs and omissions. To make their idea work, we make use of an additional data structure.

3.2 Prefix-Sum Tree

In our implementation of the Quantum Prefix Tree, we will need to keep track of which blocks are empty and which are being used by a node. For this purpose, we will use a data-structure that is famously used to (near-optimally) solve the dynamic prefix-sum problem.

► **Definition 5.** *A prefix-sum tree is a complete rooted binary tree. Each leaf node is labelled by a value in $\{0, 1\}$, and each internal node is labelled by the number of 1-valued leaf nodes descending from it.*

Let $F \subseteq [\ell]$ for ℓ a power of 2. We use $P(F)$ to denote the prefix-sum tree where the i^{th} leaf node of the tree is labelled by 1 iff $i \in F$.

A prefix-sum tree $P(F)$ will be represented in memory by an array of $\ell - 1$ blocks of memory, holding the labels of the inner nodes of $P(F)$, followed by ℓ bits, holding the labels of the leaf nodes. The blocks appear in the same order as a breadth-first traversal of $P(F)$. Consequently, for every $F \subseteq \{0, 1\}^\ell$ there is corresponding binary string of length $(\ell - 1) \log \ell + \ell$ that uniquely describes $P(F)$.

We will overload notation, and use $P(F)$ to denote this binary string of length $(\ell - 1) \log \ell + \ell$.

Allocating and deallocating

The idea now is to use the prefix tree as a memory allocator. We have $2m$ blocks of memory, and the set F will keep track of which blocks of memory are unused, or “free”.

We would then like to have an operation that allocates one of the free blocks. To implement Bernstein et al's idea, the choice of which block to allocate is made in superposition over all possible free blocks. I.e., we would like to implement the following map U_{alloc} and also its inverse, U_{free} .

$$U_{\text{alloc}} : |P(F)\rangle|0\rangle|0\rangle \rightarrow \frac{1}{\sqrt{|F|}} \sum_{i \in F} |P(F \setminus \{i\})\rangle|i\rangle|0\rangle, \quad (3)$$

The second and third registers have $O(\log m)$ bits. We do not care for what the map does when these registers are non-zero, or when $F = \emptyset$. We will guarantee that this is never the case.

Note that each internal node of the prefix tree stores the number of elements of F that are descendants to that node. In particular, the root stores $|F|$. In order to implement U_{alloc} , we then start by constructing the state

$$\frac{1}{\sqrt{|F|}} \sum_{j=1}^{|F|} |j\rangle. \quad (4)$$

While this might appear to be simple, it actually requires us to use a gate

$$U_{\text{superpose}} : |k\rangle|0\rangle \mapsto \frac{1}{\sqrt{k}} \sum_{j=1}^k |k\rangle|j\rangle. \quad (5)$$

This is much like choosing a random number between 1 and a given number k on a classical computer. Classically, such an operation cannot be done exactly if all we have at our disposition are bitwise operations (since all achievable probabilities are then dyadic rationals). Quantumly, it is impossible to implement $U_{\text{superpose}}$ efficiently without error by using only the usual set of basic gates.

So the reader should take note: it is precisely this gate which adds error to BJLM's procedure. This gate can be implemented up to distance ε using $O(\log \frac{m}{\varepsilon})$ basic gates, where m is the maximum value that k can take. I.e., using so many gates we can implement a unitary U such that the spectral norm $\|U - U_{\text{superpose}}\| \leq \varepsilon$.⁴ We will need to choose $\varepsilon \approx \frac{1}{T}$, which is the inverse of the number of times such a gate will be used throughout our algorithm.

Once we have prepared state (4), we may then use binary search, going down through the prefix tree to find out which location i corresponds to the j^{th} non-zero element of F . Using i , as we go up we can remove the corresponding child from $P(F)$, in $O(\gamma \cdot \log m)$ time, while updating the various labels on the corresponding root-to-leaf path. This requires the use of $O(\log m)$ work bits, which are $|0\rangle$ at the start and end of the operation. During this process, the register holding j is also reset to $|0\rangle$, by subtracting the element counts we encounter during the deletion process from this register. The inverse procedure U_{free} is implemented in a similar way.

3.3 Quantum Radix Tree

We may now define the quantum radix tree.

⁴ This is done by using Hadamard gates to get a superposition between 1 and the smallest power of 2 which is greater than $\frac{m}{\varepsilon}$, and then breaking this range into m equal intervals plus a remainder of size $< m$. The *remainder subspace* will have squared amplitude $\leq \varepsilon$.

10:10 Memory Compression with Quantum Random-Access Gates

► **Definition 6** (Quantum Radix Tree). Let ℓ and m be powers of 2, $S \subseteq \{0, 1\}^\ell$ be a set of size $s = |S| \leq m$, and let $R(S)$ be the classical radix tree storing S . Then, the quantum radix tree corresponding to S , denoted $|R_Q(S)\rangle$ (or $|R_Q^{\ell, m}(S)\rangle$ when ℓ and m are to be explicit), is the state

$$|R_Q(S)\rangle = \frac{1}{\sqrt{N_S}} \cdot \sum_{\tau} |R_{\tau}(S)\rangle |P(F_{\tau})\rangle,$$

where τ ranges over all injective functions $\tau : R(S) \rightarrow [2m]$ with $\tau(\text{root}) = 1$, of which there are $N_S = \frac{(2m-1)!}{(2m-|R(S)|)!}$ many, and $F_{\tau} = [2m] \setminus \tau(R(S))$ is the complement of the image of τ .

Basic operations on quantum radix trees

The basic allowed operations on a quantum radix trees are look-up and toggle, where the toggle operation is analogous to insertion and deletion in classical radix tree. Additionally, we also define a swap operation which will be used to simulate a RAG gate.

► **Lemma 7.** Let $|R_Q(S)\rangle = |R_Q^{\ell, m}(S)\rangle$ denote a quantum radix tree storing a set $S \subseteq \{0, 1\}^\ell$ of size at most m . We then define the following data structure operations.

1. **Lookup.** Given an element $e \in \{0, 1\}^\ell$, we may check if $e \in S$, so for each $b \in \{0, 1\}$, we have the map

$$|e\rangle |R_Q(S)\rangle |b\rangle \mapsto |e\rangle |R_Q(S)\rangle |b \oplus (e \in S)\rangle.$$

2. **Toggle.** Given $e \in \{0, 1\}^\ell$, we may add e to S if S does not contain e , or otherwise remove e from S . Formally,

$$|e\rangle |R_Q(S)\rangle \mapsto \begin{cases} |e\rangle |R_Q(S \cup \{e\})\rangle, & \text{if } e \notin S, \\ |e\rangle |R_Q(S \setminus \{e\})\rangle, & \text{if } e \in S. \end{cases}$$

3. **Swap.** Given an element $e \in \{0, 1\}^\ell$, $b \in \{0, 1\}$ and a quantum radix tree storing a set S , we would like **swap** to be the following map,

$$|e\rangle |R_Q(S)\rangle |b\rangle \mapsto \begin{cases} |e\rangle |R_Q(S \cup \{e\})\rangle |0\rangle, & \text{if } e \notin S \text{ and } b = 1, \\ |e\rangle |R_Q(S \setminus \{e\})\rangle |1\rangle, & \text{if } e \in S \text{ and } b = 0, \\ |e\rangle |R_Q(S)\rangle |b\rangle, & \text{otherwise.} \end{cases}$$

These operations can be implemented in worst case $O(\gamma \cdot \log m)$ time and will be error-free if we are allowed to use an error-free gate for $U_{\text{superpose}}$ (defined in Equation 5), along with other gates from set \mathcal{Q} .

Proof. Let $|b\rangle, |e\rangle$ denote the quantum states storing the elements $b \in \{0, 1\}$ and $e \in \{0, 1\}^\ell$, respectively. The data structure operations such as **lookup**, **toggle** and **swap** can be implemented reversibly in $O(\gamma \cdot \log m)$ time in the following way.

Lookup

We wish to implement the following reversible map U_{lookup} ,

$$U_{\text{lookup}} : |e\rangle |R_Q(S)\rangle |b\rangle \mapsto |e\rangle |R_Q(S)\rangle |b \oplus (e \in S)\rangle. \quad (6)$$

We do it as follows. First note that, by Definition 6,

$$|R_Q(S)\rangle = \frac{1}{\sqrt{N_S}} \sum_{\tau} |R_{\tau}(S)\rangle |P(F_{\tau})\rangle.$$

We will traverse $R_\tau(S)$ with the help of some auxiliary variables. Starting at the root node, we find the edge labeled with a prefix of e . If no such label is found then e is not present in $R_\tau(S)$. Otherwise, we traverse to the child reached by following the edge labeled by a prefix of e . Let us denote the label by L . If the child is a leaf node then terminate the process, stating that e is present in $R_\tau(S)$, else, recurse the process on e' and the tree rooted at that child node. Here e' is the binary string after removing L from e . When at some point we have determined whether $e \in S$ or not, we flip the bit b , or not. Eventually, we may conclude that $e \notin S$ before traversing the entire tree, at which point we skip the remaining logic for traversing $R_\tau(S)$ downwards (by using a control qubit). After we have traversed $R_\tau(S)$ downwards and determined whether $e \in S$, we need to undo our traversal, which we do by following the p_1 pointers (to the parent nodes) until the root is again reached, and the auxiliary variables are again set to 0.

Each comparison with the edge labels, at each traversed node, takes $O(\gamma)$ time. Hence, the entire procedure takes $O(\gamma \cdot \log m)$ time.

Toggle

Let U_{toggle} denote the following map,

$$U_{toggle} : |e\rangle|R_Q(S)\rangle \rightarrow \begin{cases} |e\rangle|R_Q(S \setminus \{e\})\rangle, & \text{if } e \in S, \\ |e\rangle|R_Q(S \cup \{e\})\rangle, & \text{if } e \notin S \end{cases} \quad (7)$$

The **toggle** operation primarily consists of two main parts: The memory allocation or de-allocation, followed by insertion or deletion, respectively.

We again traverse $R_\tau(S)$ with the help of some auxiliary variables. We start with the root node of $R_\tau(S)$, and traverse the tree downwards until we know, as above, whether $e \in S$ or not. If $e \notin S$, we will know where we need to insert nodes into $R_\tau(S)$, in order to transform it into $R_\tau(S \cup \{e\})$. Below, we will explain in detail how such an insertion must proceed. It turns out that we may need to insert either one node, or two, but never more. We may use the work qubits to compute the contents of the memory blocks that will hold this new node (or new nodes). These contents are obtained by XORing the appropriate bits of e and the appropriate parent/child pointers of the nodes we are currently traversing in the tree.

We may then use the U_{alloc} gate (once or twice) to obtain the indices of the blocks that will hold the new node(s). We then use RAG gates to swap in the contents of these blocks into memory. A fundamental and crucial detail must now be observed: the index of the memory blocks into where we inserted the new nodes is now left as part of the work qubits. This cannot be and must be dealt with, because every work bit must be again set to zero at the end of the procedure. However, a copy of this index now appears as the child pointer (p_2 or p_3) of the parents of the nodes we just created, and these pointers can thus be used to zero out the index. It is then possible to traverse the tree upwards in order to undo the various changes we did to the auxiliary variables.

If $e \in S$, on the other hand, we then do the inverse procedure. We will then know which nodes need to be removed from $R_\tau(S)$ (it will be either one or two nodes). By construction, these nodes will belong to blocks not in F_τ . We begin by setting these blocks to zero by swapping the blocks into the workspace (using the RAG gate), XORing the appropriate bits of e and the appropriate child/parent pointers so the blocks are now zero, and swapping them back. These blocks will then be set to zero, and we are left with a state akin to the right-hand side of (3). We then use the U_{free} gate to *free* the blocks, i.e., add their indices to F_τ once again. At this point we can traverse the tree upwards once more, in order to reset the auxiliary variables to zero, as required.

10:12 Memory Compression with Quantum Random-Access Gates

We now give further detail on how one must update $R_\tau(S)$ in order to insert a new element e into S . We must create a node $N := (z, p_1, p_2, p_3)$ corresponding to the element e stored at the memory location assigned by U_{alloc} procedure. Let us denote the address by k . Start with the root node of $R_\tau(S)$. If e has no common prefix with any of the labels of the root's outgoing edges, which can only happen if the root has one child, then set z to e , p_1 pointing to the root node, and, p_2 and p_3 set to 0. Moreover, set the value of the root's p_2 pointer to k if node N ends up as the left child to the root, else set root's p_3 pointer to k . In the case when e has a common prefix with one of the labels of the root's outgoing edges, let us denote the label by L and the child node by C , then further two scenarios arise: Either label L is completely contained in e , which if is the case then we traverse the tree down and run the insertion procedure recursively on e' (which is e after removing the prefix L) with the new root set C . In the case where label L is not completely contained in e , we create an internal node N' with its z variable set to the longest common prefix of e and L (which we denote by L'), p_1 pointing to root, p_2 pointing to C and p_3 pointing to N (or vice versa depending on whether node N gets to be the right or the left child). We run the U_{alloc} procedure again to get a memory location to store N' . Having done that, we now change the z value of node C to be the prefix of L after L' , and the p_1 value of node C to be the memory location of N' . Additionally, we also set z of node N to be e' , the suffix of e after L' , and we let p_1, p_2, p_3 to be, respectively, a pointer to N' , 0 and 0.

Each step in the traversal takes time $O(\gamma)$, for a total time of $O(\gamma \cdot \log m)$.

The procedure to update $R_\tau(S)$ in order to delete an element e from S is analogous to the insertion procedure mentioned above, which also can be implemented in $O(\gamma \cdot \log m)$ time.

Swap

Let U_{swap} denote the following map,

$$U_{\text{swap}} : |e\rangle |R_Q(S)\rangle |b\rangle \mapsto \begin{cases} |e\rangle |R_Q(S \cup \{e\})\rangle |0\rangle, & \text{if } e \notin S \text{ and } b = 1, \\ |e\rangle |R_Q(S \setminus \{e\})\rangle |1\rangle, & \text{if } e \in S \text{ and } b = 0, \\ |e\rangle |R_Q(S)\rangle |b\rangle, & \text{otherwise.} \end{cases}$$

To implement U_{swap} , we first run the U_{lookup} on the registers $|e\rangle$, $|R_Q(S)\rangle$ and $|b\rangle$. Conditional on the value of register $|b\rangle$ (i.e., when $b = 1$), we run U_{toggle} on the rest of the registers. We then run U_{lookup} again to attain the desired state. To summarize, the unitary $U_{\text{swap}} = U_{\text{lookup}} \cdot C_{\text{toggle}} \cdot U_{\text{lookup}}$, where C_{toggle} is controlled version of U_{toggle} (as per Lemma 3). Thus, the **swap** procedure takes a total time of $O(\gamma \cdot \log m)$. ◀

An error-less, efficient implementation of the unitary $U_{\text{superpose}}$ is impossible by using only the usual sets of basic gates. Furthermore, it is unreasonable to expect to have an error-free $U_{\text{superpose}}$ at our disposal. However, as we explained in page 9, there is a procedure to implement $U_{\text{superpose}}$ using gates from the gate set $\mathcal{B} = \{\text{CNOT}, H, S, T\}$ up to spectral distance ϵ , using only $O(\log \frac{m}{\epsilon})$ gates.

► **Corollary 8.** *Let $|R_Q(S)\rangle = |R_Q^{\ell, m}(S)\rangle$ denote a quantum radix tree storing a set $S \subseteq \{0, 1\}^\ell$ of size at most m . The data structure operations **look-up**, **toggle** and **swap**, as defined in the statement of Lemma 7 can be implemented in $O(\gamma \cdot \log \frac{m}{\epsilon})$ time and ϵ probability of error using gates from the gate set \mathcal{Q} . Here γ is the number of gates required from set \mathcal{Q} to do various basic operations on a logarithmic number of qubits.*

3.4 The simulation

Recall from Section 2.3 that we take γ to be the number of gates required to do various basic operations on a logarithmic number of qubits. In our use below, it never exceeds $O(\log M)$.

► **Theorem 9.** *Let $T, W, m < M = 2^\ell$ be natural numbers, with M and m both powers of 2, and let $\varepsilon \in [0, 1/2)$. Suppose we are given an m -sparse QRAM algorithm using time T , W work qubits and M memory qubits, that computes a Boolean relation F with error ε .*

Then we can construct a QRAM algorithm which computes F with error $\varepsilon' > \varepsilon$, and runs in time $O(T \cdot \log(\frac{T}{\varepsilon' - \varepsilon}) \cdot \gamma)$, using $W + O(\log M)$ work qubits and $O(m \log M)$ memory qubits.

Proof. Let $\mathcal{C} = (n, T, W, M, C_1, \dots, C_T)$ be the circuit of the given m -sparse QRAM algorithm computing a relation F with error ε and, let the state of the algorithm at every time-step t , when written in the computational basis be

$$|\psi_t\rangle = \sum_{u \in \{0,1\}^w} \sum_{v \in \binom{[M]}{\leq m}} \alpha_{u,v}^{(t)} \cdot \underbrace{|u\rangle}_{W \text{ qubits}} \otimes \underbrace{|v\rangle}_{M \text{ qubits}} \quad (8)$$

where the set $\binom{[M]}{\leq m}$ denotes all vectors $v \in \{0,1\}^M$ such that $|v| \leq m$. Using the description of \mathcal{C} and the fact that this algorithm is m -sparse we will now construct another QRAM algorithm \mathcal{C}' with the promised bounds. The algorithm \mathcal{C}' will have $w' = W + O(\log M)$ work bits, and $O(m \log M)$ memory bits. The memory is to be interpreted as an instance $|R_Q(S)\rangle$ of the quantum radix tree described above. Then $|v\rangle$ will be represented by the quantum radix tree $|R_Q(S_v)\rangle$, where $S_v = \{i \in [M] \mid v_i = 1\}$ is the set of positions where $v_i = 1$, so that each position $i \in [M]$ is encoded using a binary string of length ℓ .

The simulation is now simple to describe. First, the quantum radix tree is initialized. Then, each non-RAG instruction $C_i \in \mathcal{C}$ operating on the work qubits of \mathcal{C} is applied in the same way in \mathcal{C}' to same qubits among the first W qubits of \mathcal{C}' . Each RAG instruction, on the other hand, is replaced with the U_{swap} operation, applied to the the quantum radix tree. The extra work qubits of \mathcal{C}' are used as ancillary for these operations, and we note that they are always returned to zero.

If we assume that the U_{swap} operation can be implemented without error, we then have a linear-space isomorphism between the two algorithms' memory space, which maps the state $|\psi_t\rangle$ of \mathcal{C} at each time step t to the state $|\phi_t\rangle$ of \mathcal{C}' after t simulated steps:

$$|\phi_t\rangle = \sum_{u,v} \alpha_{u,v}^{(t)} \cdot \underbrace{|u\rangle}_W \otimes \underbrace{|0\rangle}_{O(\log M)} \otimes \underbrace{|R_Q(S_v)\rangle}_{O(m \log M)}.$$

Thus, if U_{swap} could be implemented without error, we could have simulated \mathcal{C} without additional error. Otherwise, as per Corollary 8, we may implement the U_{swap} unitary with an error parameter $\Omega(\frac{\varepsilon' - \varepsilon}{T})$, resulting in a total increase in error of $\varepsilon' - \varepsilon$, and an additional time cost of $O(T \log \frac{T}{\varepsilon' - \varepsilon})$. ◀

4 Simplifications of previous work

It is possible to use our main theorem to simplify the presentation of the following three results: Ambainis' Quantum Walk algorithm for solving the k -Element Distinctness problem [2], Aaronson et al's Quantum algorithms for the Closest Pair problem (CP), and the authors' previous paper on Fine-Grained Complexity via Quantum Walks [4].

All these results use quantum walk together with complicated, space-efficient, history-independent data structures. As we will see, it is possible to replace these complicated data structures with simple variants of the prefix-sum tree (Section 3.2), where the memory use is sparse, and then invoke the main theorem of our paper.

The proofs are omitted in the main body due to space constraints are instead included in the appendix.

References

- 1 Scott Aaronson, Nai-Hui Chia, Han-Hsuan Lin, Chunhao Wang, and Ruizhe Zhang. On the quantum complexity of closest pair and related problems. In *Proceedings of the 35th Computational Complexity Conference, CCC '20*, Dagstuhl, DEU, 2020. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. doi:10.4230/LIPIcs.CCC.2020.16.
- 2 A. Ambainis. Quantum walk algorithm for element distinctness. In *45th Annual IEEE Symposium on Foundations of Computer Science*, pages 22–31, 2004. doi:10.1109/FOCS.2004.54.
- 3 Daniel J. Bernstein, Stacey Jeffery, Tanja Lange, and Alexander Meurer. Quantum algorithms for the subset-sum problem. In Philippe Gaborit, editor, *Post-Quantum Cryptography*, pages 16–33, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.
- 4 Harry Buhrman, Bruno Loff, Subhasree Patro, and Florian Speelman. Limits of quantum speed-ups for computational geometry and other problems: Fine-grained complexity via quantum walks, 2021. arXiv:2106.02005.
- 5 Stacey Jeffery. *Frameworks for Quantum Algorithms*. PhD thesis, University of Waterloo, 2014.
- 6 Greg Kuperberg. A subexponential-time quantum algorithm for the dihedral hidden subgroup problem. *SIAM J. Comput.*, 35(1):170–188, July 2005. doi:10.1137/S0097539703436345.
- 7 María Naya-Plasencia and André Schrottenloher. Optimal merging in quantum -xor and -xor-sum algorithms. In *Advances in Cryptology – EUROCRYPT 2020: 39th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zagreb, Croatia, May 10–14, 2020, Proceedings, Part II*, pages 311–340, Berlin, Heidelberg, 2020. Springer-Verlag. doi:10.1007/978-3-030-45724-2_11.

A Simplifications of previous work (the proofs)

A.1 Ambainis’ Walk Algorithm for Element Distinctness

Ambainis’ description and analysis of his data structure is complicated, and roughly 6 pages long, whereas a presentation of his results with a simple data structure and an appeal to our theorem requires less than 2 pages, as we will now see. Also, the presentation of the algorithm is considerably muddled by the various difficulties and requirements pertaining to the more complicated data structure. In a presentation of his results that would then appeal to Theorem 1, we have a very clear separation of concerns.

Ambainis’ algorithm is a $\tilde{O}(n^{\frac{k}{k+1}})$ -time solution to the following problem:

► **Definition 10** (*k*-Element Distinctness). *Given a list L of n integers in Σ are there k elements $x_{i_1}, \dots, x_{i_k} \in L$ such that $x_{i_1} = \dots = x_{i_k}$.*

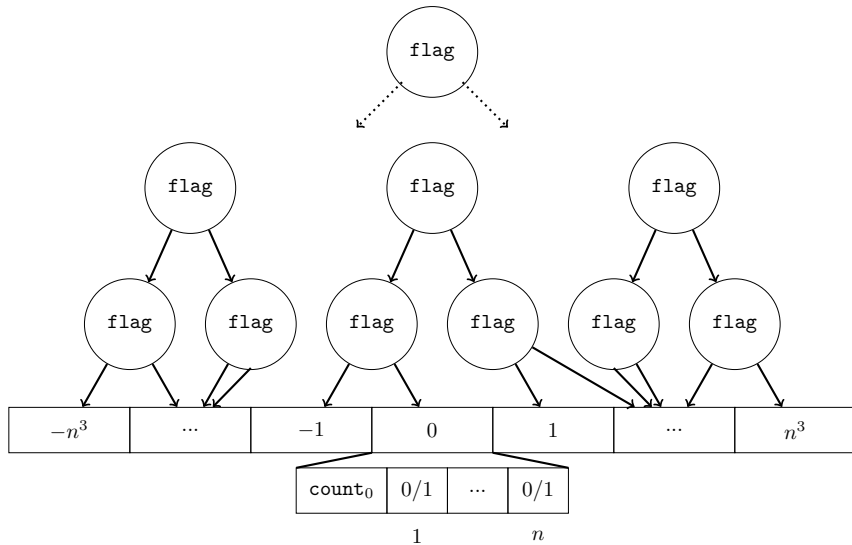
Ambainis’ algorithm for *k*-Element Distinctness [2] is quantum walk algorithm on a Johnson graph $J(n, r)$ with $r = n^{k/k+1}$ and runs in $\tilde{O}(n^{k/k+1})$ time. The crucial ingredient in making the algorithm time efficient is the construction of data-structure which can store a set $S \subseteq [n] \times \Sigma$ of elements of size r , under efficient insertions and removals, so that one

may efficiently query at any given time whether there exist k elements $(i_1, x_1), \dots, (i_k, x_k)$ in S with distinct indices i_1, \dots, i_k but equal labels $x_1 = \dots = x_k$. Ambainis makes use of skip-lists and hash tables, ensuring that all operations run in $O(\log^4(n + |\Sigma|))$ time. However, if one does not care about space-efficiency, there is a much simpler data structure that serves the same purpose. The following definition is illustrated in Figure 2.

► **Definition 11.** Let $S \subseteq [n] \times \Sigma$, with $|S| = r$ and $|\Sigma| = n^{O(1)}$ a power of 2, and such that every $i \in [n]$ appears in at most one pair $(i, x) \in S$. The k -element-distinctness tree that represents S , denoted $T_k(S)$, is a complete rooted binary tree with $|\Sigma|$ leaves. Each leaf node $x \in \Sigma$ is labeled by a bit vector $B_x \in \{0, 1\}^n$ and a number $\text{count}_x \in \{0, \dots, n\}$, so that $B_x[i] = 1$ iff $(i, x) \in S$, and the count_x is the Hamming weight of B_x . Each internal node w is labeled by a bit $\text{flag}_w \in \{0, 1\}$ which indicates whether there exists a leaf x , descendent of w , with $\text{count}_x \geq k$.

Memory Representation

A k -element-distinctness tree is represented in the memory by an array of $|\Sigma| - 1$ bits of memory holding the flags of the internal nodes, followed by $|\Sigma|$ blocks of $n + \lceil \log n \rceil$ bits of memory each, holding the labels of the leaf nodes. The blocks appear in the same order as a breadth-first traversal of $T_k(S)$. Consequently, for every $S \subseteq [n] \times \Sigma$ there is a corresponding binary string of length $|\Sigma| - 1 + (n + \lceil \log n \rceil)|\Sigma|$ that uniquely encodes $T_k(S)$. Crucially, if $|S| = r$, then at most $O(r(\log \Sigma + \log n))$ of these bits are 1. So for $|\Sigma| = \text{poly}(n)$, the encoding is $\tilde{O}(r)$ -sparse.



■ **Figure 2** Data structure for the k -Element Distinctness problem.

Implementation of data structure operations

It is clear from the definition of k -element-distinctness tree and its memory representation that a tree $T_k(S)$ represents a set $S \subseteq [n] \times \Sigma$ in a history-independent way. We will now argue that all the required data structure operations take $O(\log n)$ time in the worst case. Let (i, x) denote an element in $[n] \times \Sigma$.

10:16 Memory Compression with Quantum Random-Access Gates

- **Insertion.** To insert (i, x) in the tree, first increase the value of the count variable of the leaf x , and set $B_x[i] = 1$. Then, if $\text{count} \geq k$, set $\text{flag}_w = 1$ for all w on the root-to- x path. This update requires $O(\log n)$ time as $|\Sigma| = \text{poly}(n)$.
- **Deletion.** The procedure to delete is similar to the insertion procedure. To delete (i, x) in the tree, first decrease the value of the count_x and set $B[i] = 0$. If $\text{count} < k$, then, for all w on the root-to- x path which do not have both children w_0, w_1 with $\text{flag}_{w_0} = \text{flag}_{w_1} = 1$, set $\text{flag}_w = 0$. This requires $O(\log n)$ time.
- **Query.** To check if the tree has k distinct indices with the same x , we need only check if $\text{flag}_{\text{root}} = 1$, which takes $O(1)$ time.

Runtime, error and memory usage

Using the above data-structure, the runtime of Ambainis' algorithm is now $\tilde{O}(n^{\frac{k}{k+1}})$ time. The total memory used is $O(n|\Sigma|)$ bits. However, note that at any point of time in any branch of computation Ambainis' walk algorithm stores sets of size $r = O(n^{\frac{k}{k+1}})$. Hence their algorithm with this data structure is a $\tilde{O}(n^{\frac{k}{k+1}})$ -sparse algorithm. Thus, invoking Theorem 1 we conclude the following.

► **Corollary 12.** *There is a bounded-error QRAM algorithm that computes k -Element Distinctness in $\tilde{O}(n^{k/k+1})$ time using $\tilde{O}(n^{k/k+1})$ memory qubits.*

A.2 Quantum Algorithms for Closest-Pair and related Problems

The paper of Aaronson et al [1] provides quantum algorithms and conditional lower-bounds for several variants of the Closest Pair problem (CP).

Let $\Delta(a, b) = \|a - b\|$ denote the Euclidean distance. We then describe the Closest Pair problem under Euclidean distance Δ , but we could have chosen any other metric Δ in d -dimensional space which is strongly-equivalent to the Euclidean distance (such as ℓ_p distance, Manhattan distance, ℓ_∞ , etc).

► **Definition 13** (Closest Pair $\text{CP}(n, d)$ problem). *In the $\text{CP}(n, d)$ problem, we are given a list P of n distinct points in \mathcal{R}^d , and wish to output a pair $a, b \in P$ with the smallest $\Delta(a, b)$.*

We may also define a threshold version of CP.

► **Definition 14.** *In the $\text{TCP}(n, d)$ problem, we are given a set $P = \{p_1, \dots, p_n\}$ of n points in \mathbb{R}^d and a threshold $\varepsilon \geq 0$, and we wish to find a pair of points $a, b \in P$ such that $\Delta(a, b) \leq \varepsilon$, if such a pair exists.*

For simplicity, so we may disregard issues of representation of the points, we assume that all points are specified using $O(\log n)$ bits of precision. By translation, we can assume that all the points lie in the integer hypercube $[L]^d$ for some $L = \text{poly}(n)$, and that $\delta \in [L]$, also.

It is then possible to solve CP by running a binary search over the (at most n^2) different values of $\delta \in \{\Delta(p_i, p_j) \mid i, j \in [n]\}$ and running the corresponding algorithm for TCP. This will add an additional $O(\log n)$ factor to the running time.

The $\text{TCP}(n, d)$ problem is a query problem with certificate complexity 2. If one is familiar with quantum walks, it should be clear that we may do a quantum walk on the Johnson graph over n vertices, to find a pair with the desired property, by doing $O(n^{2/3})$ queries to the input. Again, if one is familiar with quantum walks, one will realize that, in order to implement this walk efficiently, we must dynamically maintain a set $S \subseteq [n]$, and at each step in the quantum walk, we must be able to add or remove an element i to S , and answer a query of the form: does there exist a pair $i, j \in S$ with $\Delta(p_i, p_j) \leq \varepsilon$?

The only difficulty, now, is to implement an efficient data structure that can dynamically maintain S in this way, and answer the desired queries, while being time and space efficient. Aaronson et al construct a data-structure which can store a set $S \subseteq [n] \times [L]^d$ of points of size r , under efficient insertions and removals, so that one may query at any given time whether there exist two points in S which are ε -close. They do so by first discretizing $[L]^d$ into a hypergrid of width ε/\sqrt{d} , as explained below, and then use a hash table, skip list, and a radix tree to maintain the locations of the points in the hypergrid.

The presentation of the data structure in the paper is roughly 6 pages long, and one must refer to Ambainis' paper for the error analysis, which is absent from the paper. As we will see, a simple, sparse data structure for the same purpose can be described in less than 2 pages, and then an appeal to Theorem 1 gives us the same result up to log factors.

Discretization

We discretize the cube $[L]^d$ into a hypergrid of width $w = \frac{\varepsilon}{\sqrt{d}}$, and let $\text{id}(p)$ denote the box containing p in this grid. I.e., we define a function $\text{id}(p) : [L]^d \rightarrow \{0, 1\}^{\lceil d \log(L/\varepsilon) \rceil}$ by

$$\text{id}(p) = (\lfloor p(1)/w \rfloor, \dots, \lfloor p(d)/w \rfloor) \quad (\text{represented in binary}).$$

Let $\Sigma = \{0, 1\}^{\lceil d \log(L/\varepsilon) \rceil}$ denote the set of all possible boxes. We say that two boxes $g, g' \in \Sigma$ are neighbours if

$$\sqrt{\sum_{i=1}^d \|g(i) - g'(i)\|^2} \leq \sqrt{d}.$$

A loose estimate will show there can be at most $(2\sqrt{d} + 1)^d$ neighbours for any box. This method of discretization ensures the following crucial property:

- **Observation 15** (Observation 45 [1]). *Let p, q be any two distinct points in $[0, L]^d$, then*
1. *if $\text{id}(p) = \text{id}(q)$, then $\Delta(p, q) \leq \varepsilon$, and*
 2. *if $\Delta(p, q) \leq \varepsilon$, then $\text{id}(p)$ and $\text{id}(q)$ are neighbours.*

From Observation 15, it follows that $i, j \in [n]$ exist with $\Delta(p_i, p_j) \leq \varepsilon$, if and only if we have one of the following two cases:

- Either there is such a pair i, j with $\text{id}(p_i) = \text{id}(p_j)$.
- Or there is no such pair, and then there must exist two neighbouring boxes $\text{id}(i)$ and $\text{id}(j)$, each containing a single point, with $\Delta(p_i, p_j) \leq \varepsilon$.

We now describe the data structure itself. Let us assume without loss of generality that n is a power of 2.

► **Definition 16** (Data Structure for CP). *Let $S \subseteq [n] \times \Sigma$, with $|S| = r$, and such that every $i \in [n]$ appears in at most one pair $(i, x) \in S$. The closest-pair tree that represents S , denoted by $T_{CP}(S)$, is a complete rooted binary tree with $|\Sigma|$ leaves. Each leaf node $x \in \Sigma$ is labeled by a number $\text{external}_x \in \{0, \dots, n\}$, and a prefix-sum tree $P(S_x)$ representing the set $S_x = \{i \in [n] \mid (i, x) \in S\}$. Each internal node w is labeled by a bit $\text{flag}_w \in \{0, 1\}$. These labels obey the following rules:*

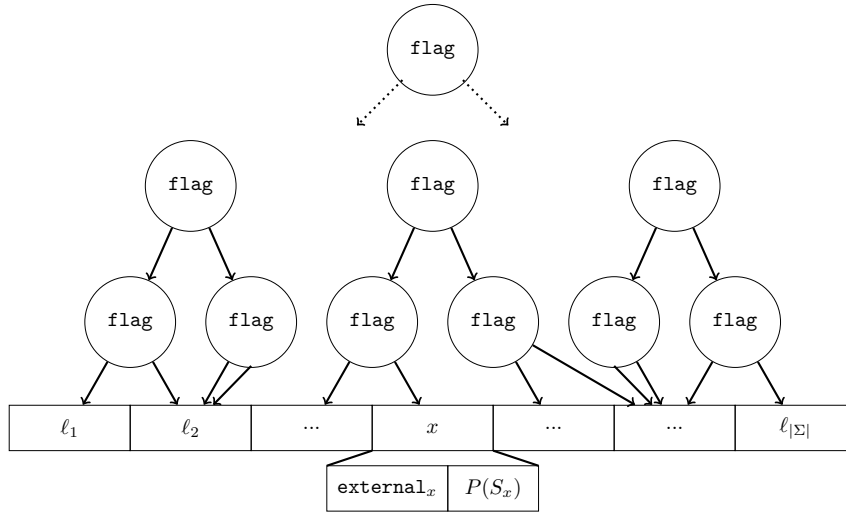
- *If $|S_x| = 1$, then external_x is the number of boxes $y \neq x$, which are neighbours of x , and which have $|S_y| = 1$ and $\Delta(p_i, p_j) \leq \varepsilon$ for the (unique) $j \in S_y$.*
- *If $|S_x| \geq 2$, then $\text{external}_x = 0$.*
- *The $\text{flag}_w = 1$ if any of the children x descendants to the internal node w have either $|S_x| \geq 2$ or $|S_x| = 1$ and $\text{external}_x \geq 1$.*

10:18 Memory Compression with Quantum Random-Access Gates

It follows from the above discussion that there exist two elements $(i, x), (j, y) \in S$ with $\Delta(p_i, p_j) \leq \varepsilon$ if and only if $\mathbf{flag}_{\text{root}} = 1$ in $T_{CP}(S)$. We now show how to efficiently maintain $T_{CP}(S)$ under insertions and removals.

Memory Representation

A TCP tree is represented in the memory by an array of $|\Sigma| - 1$ bits of memory holding the flags of the internal nodes, followed by $|\Sigma|$ blocks of $n \log n + n$ bits of memory each, holding the labels of the leaf nodes. The blocks appear in the same order as a breadth-first traversal of $T_{CP}(S)$. Consequently, for every $S \subseteq [n] \times \Sigma$ there is a corresponding binary string of $|\Sigma| - 1 + (n \log n + n)|\Sigma|$ that uniquely encodes $T_{CP}(S)$. Crucially, if $|S| = r$, then at most $O(r(\log |\Sigma| + \log n))$ of these bits are 1. Since $|\Sigma| = L^{O(d)} = \text{poly}(n)$ (recall $d = O(1)$), the encoding is $\tilde{O}(r)$ -sparse.



■ **Figure 3** Data structure for the CP problem.

Implementation of data structure operations

It is clear from the definition of TCP tree and its memory representation that a tree $T_{CP}(S)$ represents a set $S \subseteq [n] \times \Sigma$ in a history-independent way. We will now argue that all the required data structure operations take $O(\log n)$ time in the worst case. For every $(i, x) \in [n] \times [L]^d$ there is a corresponding $(i, z) \in [n] \times \Sigma$, with $z = \text{id}(x)$, stored in the data structure.

- **Insertion.** To insert (i, x) in the tree, first go to the memory location corresponding to leaf x . Begin by inserting i in the prefix-sum tree $P(S_x)$. Then three cases arise
 - If $|S_x| = 1$ then for every neighbour y of x with $|S_y| = 1$ do the following: Using the prefix-sum tree at leaf y obtain the only non-zero leaf index j of $P(S_y)$. This operation takes $\log n$ time. Then check if $\Delta(p_i, p_j) \leq \varepsilon$, if yes then increase the values of both $\mathbf{external}_x$ and $\mathbf{external}_y$ by 1. If this caused $\mathbf{external}_y > 0$ then set $\mathbf{flag}_w = 1$ for all internal nodes w on the path from leaf y to the root of $T_{CP}(S)$.
 After going over all neighbours, check if $\mathbf{external}_x \geq 1$, if it is then set $\mathbf{flag}_w = 1$ for all internal nodes w on the path from leaf x to the root of $T_{CP}(S)$. This process takes at most $(2\sqrt{d} + 1)^d \log n$ time as there will be at most $(2\sqrt{d} + 1)^d$ neighbours, which is $O(\log n)$ for $d = O(1)$.

- If $|S_x| = 2$ using the prefix-sum tree $P(S_x)$ obtain the only other non-zero leaf index $i' \neq i$ of $P(S_x)$. Then for all neighbours y of x with $|S_y| = 1$ do the following: Using the prefix-sum tree $P(S_y)$ obtain the only non-zero index j of $P(S_y)$. Check if $\Delta(p_{i'}, p_j) \leq \varepsilon$, and if so decrease the value of $\mathbf{external}_y$ by 1. If that results in making $\mathbf{external}_y = 0$ then set $\mathbf{flag}_w = 0$ for the parent of y , unless the other child y' of the parent of y has $|S_{y'}| \geq 2$ or $\mathbf{external}_{y'} \geq 1$. Likewise, among all the internal nodes w that are on the path from the root to y 's parent, update the \mathbf{flag}_w accordingly, i.e., set $\mathbf{flag}_w = 1$ if any child u of w has $\mathbf{flag}_u = 1$, and otherwise set $\mathbf{flag}_w = 0$. Having done that, set $\mathbf{external}_x = 0$ and set $\mathbf{flag}_w = 1$ for all internal nodes w from leaf x to the root $T_{CP}(S)$. This process also takes $O(\log n)$ time (when d is a constant).
 - If $|S_x| > 2$ then do nothing.
- **Deletion.** The procedure to delete is similar to the insertion procedure.
- **Query.** To check if the tree has a pair $(i, x), (j, y) \in S$ such that $\Delta(p_i, p_j) \leq \varepsilon$, we need only check if $\mathbf{flag}_{\text{root}} = 1$, which takes $O(1)$ time.

Runtime, error and memory usage

Using the above data-structure, the runtime of this TCP algorithm is now $\tilde{O}(n^{\frac{2}{3}})$ time. The total memory used is $\tilde{O}(n|\Sigma|)$ bits. However, note that at any point of time in any branch of computation this algorithm stores sets of size $r = O(n^{\frac{2}{3}})$. Hence their algorithm with this data structure is a $\tilde{O}(n^{\frac{2}{3}})$ -sparse algorithm. Thus, invoking Theorem 1 we conclude the following.

► **Corollary 17.** *There is a bounded-error QRAM algorithm that computes TCP in $\tilde{O}(n^{2/3})$ time using $\tilde{O}(n^{2/3})$ memory qubits.*

A.3 Fine-Grained Complexity via Quantum Walks

The authors' own paper [4] shows that the quantum 3SUM conjecture, which states that there exists no truly sublinear quantum algorithm for 3SUM, implies several other quantum lower-bounds. The reductions use quantum walks together with complicated space-efficient data structures. We had already realized, when writing the paper, that simple yet space-inefficient data structures could be used instead, and included this observation in the paper, so we will not repeat it here. Section 3.1, with the space inefficient sparse data structures, is 4 pages long, whereas section 3.2, with the complicated space efficient data structures, is 12 pages long.

Quantum Speedups for Treewidth

Vladislavs Kļevickis ✉

Centre for Quantum Computer Science, Faculty of Computing, University of Latvia, Riga, Latvia

Krišjānis Prūsis ✉

Centre for Quantum Computer Science, Faculty of Computing, University of Latvia, Riga, Latvia

Jevgēnijs Vihrovs ✉ 

Centre for Quantum Computer Science, Faculty of Computing, University of Latvia, Riga, Latvia

Abstract

In this paper, we study quantum algorithms for computing the exact value of the treewidth of a graph. Our algorithms are based on the classical algorithm by Fomin and Villanger (Combinatorica 32, 2012) that uses $O(2.616^n)$ time and polynomial space. We show three quantum algorithms with the following complexity, using QRAM in both exponential space algorithms:

- $O(1.618^n)$ time and polynomial space;
- $O(1.554^n)$ time and $O(1.452^n)$ space;
- $O(1.538^n)$ time and space.

In contrast, the fastest known classical algorithm for treewidth uses $O(1.755^n)$ time and space. The first two speed-ups are obtained in a fairly straightforward way. The first version uses additionally only Grover's search and provides a quadratic speedup. The second speedup is more time-efficient and uses both Grover's search and the quantum exponential dynamic programming by Ambainis et al. (SODA '19). The third version uses the specific properties of the classical algorithm and treewidth, with a modified version of the quantum dynamic programming on the hypercube. As a small side result, we give a new classical time-space tradeoff for computing treewidth in $O^*(2^n)$ time and $O^*(\sqrt{2^n})$ space.

2012 ACM Subject Classification Theory of computation → Parameterized complexity and exact algorithms; Theory of computation → Quantum computation theory

Keywords and phrases Quantum computation, Treewidth, Exact algorithms, Dynamic programming

Digital Object Identifier 10.4230/LIPIcs.TQC.2022.11

Related Version *Preprint*: <https://arxiv.org/abs/2202.08186>

Funding Supported by the project “Quantum algorithms: from complexity theory to experiment” funded under ERDF programme 1.1.1.5.

1 Introduction

For many NP-complete problems, the exact solution can be found much faster than a brute-force search over the possible solutions; it is not so rare that the best currently known algorithms are exponential [9]. Perhaps one of the most famous examples is the travelling salesman problem, where a naive brute-force requires $O^*(n!)$ computational time, but a dynamic programming algorithm solves it exactly only in $O^*(2^n)$ time [4, 14]. Such algorithms are studied also because they can reveal much about the mathematical structure of the problem and because sometimes in practice they can be more efficient than subexponential algorithms with a large constant factor in their complexity.

With the advent of quantum computing, it is curious how quantum procedures can be used to speed up such algorithms. A clear example is illustrated by the SAT problem: while iterating over all possible assignments to the Boolean formula on n variables gives $O^*(2^n)$ time, Grover's search [13] can speed this up quadratically, resulting in $O^*(\sqrt{2^n})$ time. Grover's search can also speed up exponential dynamic programming: recently Ambainis et al. [1] have shown how to apply Grover's search recursively together with classical



© Vladislavs Kļevickis, Krišjānis Prūsis, and Jevgēnijs Vihrovs;
licensed under Creative Commons License CC-BY 4.0

17th Conference on the Theory of Quantum Computation, Communication and Cryptography (TQC 2022).

Editors: François Le Gall and Tomoyuki Morimae; Article No. 11; pp.11:1–11:18

Leibniz International Proceedings in Informatics



LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

precalculation to speed up the $O^*(2^n)$ dynamic programming introduced by Bellman, Held and Karp [4, 14] to a $O(1.817^n)$ quantum algorithm. For some problems like the travelling salesman problem and minimum set cover, the authors also gave a more efficient $O(1.728^n)$ time quantum algorithm by combining Grover's search with both divide & conquer and dynamic programming techniques. Their approach has been subsequently applied to find a speedup for more NP-complete problems, including graph coloring [19], minimum Steiner tree [18] and optimal OBDD ordering [20].

In this paper, we focus on the NP-complete problem of finding the treewidth of a graph. Informally, the treewidth is a value that describes how close the graph is to a tree; for example, the treewidth is 1 when the graph is a tree, while the treewidth of a complete graph on n vertices is $n - 1$. This quantity is prominently used in parameterized algorithms, as many problems are efficiently solvable when treewidth is small, such as vertex cover, independent set, dominating set, Hamiltonian cycle, graph coloring, etc. [3]. The applications of treewidth, both theoretical and practical, are numerous, see [5] for a survey. If the treewidth is at most k , it can be computed exactly in $O(n^{k+2})$ time [2]; 2-approximated in parameterized linear time $2^{O(k)}n$ [17]; $O(\sqrt{\log k})$ -approximated in polynomial time [8]; k -approximated in $O(k^7 n \log n)$ time [10].

As for exact exponential time treewidth algorithms, both currently most time and space efficient algorithms were proposed by Fomin and Villanger in [11]: the first uses $O(1.755^n)$ time and space and the second requires $O(2.616^n)$ time and polynomial space. The crucial ingredient of these algorithms is a combinatorial lemma that upper bounds the number of connected subsets with fixed neighborhood size (Lemma 7), as well as gives an algorithm that lists such sets.

Our main motivation for tackling these algorithms is that although the $O(1.817^n)$ quantum algorithm from [1] is applicable to treewidth, it is still less efficient than Fomin's and Villanger's. In this paper we show that their techniques are also amenable to quantum search procedures. In particular, we focus on their polynomial space algorithm. This algorithm has two nested procedures: the first procedure uses Lemma 7 to search through specific subsets of vertices S to fix as a bag of the tree decomposition; the second procedure finds the optimal width of the tree decomposition with S as a bag.

We find that Grover's search can be applied to the listing procedure of Lemma 7, thus speeding up the first procedure quadratically. For the second procedure, classically one can use either the $O^*(2^n)$ time and space dynamic programming algorithm or the $O^*(4^n)$ time and polynomial space divide & conquer algorithm (Fomin and Villanger use the latter), which both were introduced in [6]. The divide & conquer algorithm we can also speed up using Grover's search. Thus, we obtain a quadratic speedup for the polynomial space algorithm:

► **Theorem 1.** *There is a bounded-error quantum algorithm that finds the exact treewidth of a graph on n vertices in $O(1.61713^n)$ time and polynomial space.*

Next, using the fact that the $O^*(2^n)$ dynamic programming algorithm can be sped up to an $O^*(1.817^n)$ quantum algorithm together with the quadratic speedup of Lemma 7, we obtain our second quantum algorithm:

► **Theorem 2.** *Assuming the QRAM data structure, there is a bounded-error quantum algorithm that finds the exact treewidth of a graph on n vertices in $O(1.55374^n)$ time and $O(1.45195^n)$ space.*

The last theorem suggests a possibility for an even more efficient algorithm by trading some space for time. We achieve this by proving a treewidth property which essentially states that we can precalculate some values of dynamic programming for the original graph, and reuse these values in the dynamic programming for its subgraphs (Lemma 22). This

allows us a global precalculation, which can be used in the second procedure of the treewidth algorithm. To do that, we have to modify the $O(1.817^n)$ algorithm of [1]. We refer to it as the asymmetric quantum exponential dynamic programming. This gives us the following algorithm:

► **Theorem 3.** *Assuming the QRAM data structure, there is a bounded-error quantum algorithm that finds the exact treewidth of a graph on n vertices in $O(1.53793^n)$ time and space.*

Lastly, we observe that replacing the $O^*(4^n)$ divide & conquer algorithm in the classical $O(2.616^n)$ polynomial space algorithm by the $O^*(2^n)$ dynamic programming only lowers the time complexity to $O^*(2^n)$. However, the interesting consequence is that the space requirement drops down to $O^*(\sqrt{2^n})$. Hence, we obtain a *classical* time-space tradeoff:

► **Theorem 4.** *The treewidth of a graph on n vertices can be computed in $O^*(2^n)$ time and $O^*(\sqrt{2^n})$ space.*

Time-wise, this is more efficient than the $O(2.616^n)$ time polynomial space algorithm, and space-wise, this is more efficient than the $O(1.755^n)$ time and space algorithm. It also fully subsumes the time-space tradeoffs for permutation problems proposed in [16] applied to treewidth.

2 Preliminaries

We denote the set of integers from 1 to n by $[n]$. For a set S , denote the set of all its subsets by 2^S . We call a permutation of a set of vertices $S \subseteq V$ a bijection $\pi : S \rightarrow [|S|]$. We denote the set of permutations of S by $\Pi(S)$. For a permutation $\pi \in \Pi(S)$, let $\pi_{<v} = \{w \in S \mid \pi(w) < \pi(v)\}$ and $\pi_{>v} = \{w \in S \mid \pi(w) > \pi(v)\}$. We say that a subset T is a *prefix* of $\pi : S \rightarrow [|S|]$ if $\{\pi(t) \mid t \in T\} = \{1, \dots, i\}$ for some $i \in \{1, \dots, |S|\}$ or T is an empty set. Similarly, we say that T is a *suffix* of $\pi : S \rightarrow [|S|]$ if $\{\pi(t) \mid t \in T\} = \{i, \dots, |S|\}$ for some $i \in \{1, \dots, n\}$ or T is an empty set.

We write $O(f(n)) = \text{poly}(n)$ if $f(n) = O(n^c)$ for some constant c . Let $O(\text{poly}(n)f(m)) = O^*(f(m))$. This is useful since our subprocedures will often have some running time $f(m)$ times some function that depends on the size of the input graph G on n vertices. In this paper, we are primarily concerned with the exponential complexity of the algorithms, hence, we are interested in the $f(m)$ value of an $O^*(f(m))$ complexity.

Graph notation

For a graph $G = (V, E)$ and a subset of vertices $S \subseteq V$, denote $G[S]$ as the graph induced in G on S . For a subset of vertices $S \subseteq V$, let $N(S) = \{v \in V - S \mid u \in S, \{u, v\} \in E\}$ be its *neighborhood*. We call a subset $S \subseteq V$ connected if $G[S]$ is connected, and $C \subseteq V$ a clique if $G[C]$ is a complete graph.

For completeness, we also describe the notions of *potential maximum cliques* and *minimal separators*, which are specific subsets of V . Our quantum algorithms don't additionally rely on them more than the algorithm of Fomin and Villanger; for their further properties, see e.g. [11].

A graph is called *chordal* if every cycle of length at least 4 contains an edge that connects non-consecutive vertices of the cycle. A *triangulation* of a graph $G = (V, E)$ is a chordal graph $H = (V, E')$ such that $E \subseteq E'$. A triangulation H is *minimal* if every graph obtained from H by removing any edge is not a triangulation. A set of vertices $\Omega \subseteq V$ of a graph $G = (V, E)$ is a *potential maximum clique* if there is a minimal triangulation H of G such that Ω is a maximal clique of H .

11:4 Quantum Speedups for Treewidth

For two non-adjacent vertices u and v in a graph G , a subset of vertices $S \subseteq V$ is a u, v -separator if u and v are in different connected components of $G[V - S]$. A u, v -separator is *minimal* if none of its proper subsets is a u, v -separator. A set S is called a *minimal separator*, if there exist two vertices $u, v \in G$ such that S is a minimal u, v -separator.

Treewidth

A *tree decomposition* of a graph $G = (V, E)$ is a pair (X, T) , where $T = (V_T, E_T)$ is a tree and $X = \{\chi_i \mid i \in V_T\} \subseteq 2^V$ such that:

- $\bigcup_{\chi \in X} \chi = V$;
- for each edge $\{u, v\} \in E$, there exists $\chi \in X$ such that $u, v \in \chi$;
- for any vertex $v \in V$ in G , the set of vertices $\{\chi \mid v \in \chi\}$ forms a connected subtree of T .

We call the subsets $\chi \in X$ *bags* and the vertices of T *nodes*. The *width* of (X, T) is defined as the maximum size of $\chi \in X$ minus 1. The *treewidth* of G is defined as the minimum width of a tree decomposition of G and we denote it by $\text{tw}(G)$. We also consider optimal tree decompositions given that some subset $\chi \in V$ is a bag of the tree. We denote the smallest width of a tree decomposition of G among those that contain χ as a bag by $\text{tw}(G, \chi)$.

Approximations

For the binomial coefficients, we use the following well-known approximation:

► **Theorem 5 (Entropy approximation).** For any $k \in [0, 1]$, we have $\binom{n}{k} \leq 2^{H(\frac{k}{n}) \cdot n}$, where $H(\epsilon) = -(\epsilon \log_2(\epsilon) + (1 - \epsilon) \log_2(1 - \epsilon))$ is the binary entropy function.

Quantum subroutines

Our algorithms use a well-known variation of Grover's search, quantum minimum finding:

► **Theorem 6 (Theorem 1 in [7]).** Let $\mathcal{A} : N \rightarrow [n]$ be an exact quantum algorithm with running time T . Then there is a bounded-error quantum algorithm that computes $\min_{i \in [N]} \mathcal{A}(i)$ in $O^*(T\sqrt{N})$ time.

Two of our algorithms use the QRAM data structure [12]. This structure stores N memory entries and, given a superposition of memory indices together with an empty data register $\sum_{i \in [N]} \alpha |i\rangle |\mathbf{0}\rangle$, it produces the state $\sum_{i \in [N]} \alpha |i\rangle |\text{data}_i\rangle$ in $O(\log N)$ time. In our algorithms, N will always be exponential in n , which means that a QRAM operation is going to be polynomial in n . Thus, this factor will not affect the exponential complexity, which we are interested in.

In our algorithms, we will often have a quantum algorithm that takes exact subprocedures (like in Theorem 6), and give it bounded-error subprocedures. Since we always going to take $O(\exp(n))$ number of inputs, this issue can be easily solved by repeating the subprocedures $\text{poly}(n)$ times to boost the probability of correct answer to $1 - O(1/\exp(n))$: it can be then shown that the branch in which all the procedures have correct answers has constant amplitude. The final bounded-error algorithm incurs only a polynomial factor, and does not affect the exponential complexity. We also note that on a deeper perspective, all our quantum subroutines are based on the primitive of Grover's search [13]; an implementation of Grover's search with bounded-error inputs that does not incur additional factors in the complexity has been shown in [15].

We also are going to encounter an issue that sometimes we have some real parameter $\alpha \in [0, 1]$ and we are examining $\binom{n}{\alpha n}$. Since αn is not integer, this value is not defined; however, we can take this to be any value between $\binom{n}{\lfloor \alpha n \rfloor}$ or $\binom{n}{\lceil \alpha n \rceil}$, as they differ only by a factor of n . Thus, this does not produce an issue for the exponential complexity analysis. Henceforward we abuse the notation and simply write $\binom{n}{\alpha n}$.

3 Combinatorial lemma

In this section we describe how the main combinatorial lemma of [11] can be sped up quantumly quadratically using Grover's search.

► **Lemma 7** (Lemmas 3.1. and 3.2. in [11]). *Let $G = (V, E)$ be a graph. For every $v \in V$ and $b, f \geq 0$, the number of connected subsets $B \subseteq V$ such that (1) $v \in B$, (2) $|B| = b + 1$, and (3) $|N(B)| = f$ is at most $\binom{b+f}{b}$. There also exists an algorithm that lists all such sets in $O^*\left(\binom{b+f}{b}\right)$ time and polynomial space.*

Informally, this lemma is used in the treewidth algorithm to search for a set, such that, if fixed as a bag of the tree decomposition, the remaining graph breaks down into connected components of bounded size; then, the optimal width of the tree decomposition with this bag fixed can be solved using algorithms from Section 4. The lemma upper bounds the number of sets to consider.

Their proof of this lemma (see Appendix A) essentially gives a branching algorithm that splits the problem into several problems of the same type, and solves them recursively. The idea for applying Grover's search to such a branching algorithm is simple. The algorithm that generates all sets can be turned into a procedure that, given a number $i \in \left[\binom{b+f}{b}\right]$ of the set we need to generate, generates this set in polynomial time. Then, we can run Grover's search over all integers in $\left[\binom{b+f}{b}\right]$ on this procedure.

► **Lemma 8.** *Let $G = (V, E)$ be a graph on n vertices, and $\mathcal{A} : 2^V \rightarrow [n]$ be an exact quantum algorithm with running time T . For every $v \in V$ and $b, f \geq 0$, let $\mathcal{B}_{v,b,f}$ be the set of connected subsets $B \subseteq V$ satisfying the conditions of Lemma 7. Then there is a bounded-error quantum algorithm that computes $\min_{B \in \mathcal{B}_{v,b,f}} \mathcal{A}(B)$ in time $O^*\left(T \sqrt{\binom{b+f}{b}}\right)$.*

4 Fixed bag treewidth algorithms

In this section we describe algorithms that calculate the optimal treewidth of a graph with the condition that a subset of its vertices is fixed as a bag of the tree decomposition. We then show ways to speed them up quantumly. Both approaches were given by Bodlaender et al. [6].

4.1 Treewidth as a linear ordering

Both of these algorithms use the fact that treewidth can be seen as a graph linear ordering problem. For a detailed description, see Section 2.2 of [6], from where we also borrow a lot of notation. We will also use the properties of this formulation in our improved quantum algorithm.

A linear ordering of a graph $G = (V, E)$ is a permutation $\pi \in \Pi(V)$. The task of a linear ordering problem is finding $\min_{\pi \in \Pi(V)} f_G(\pi)$, for some known function f_G .

For two vertices $v, w \in V$, define a predicate $P_\pi^G(v, w)$ to be true iff there is a path from v to w in G such that all internal vertices in that path are before v and w in π . Then define $R_\pi^G(v)$ to be the number of vertices w such that $\pi(w) > \pi(v)$ and $P_\pi^G(v, w)$ holds. The following proposition gives a description of treewidth as a linear ordering problem:

► **Proposition 9** (Proposition 3 in [6]). *Let $G = (V, E)$ be a graph, and k a non-negative integer. The treewidth of G is at most k iff there is a linear ordering π of G such that for each $v \in V$, we have $R_\pi^G(v) \leq k$.*

11:6 Quantum Speedups for Treewidth

For a set of vertices $S \subseteq V$ and a vertex $v \notin S$, define $Q_G(S, v) = \{w \in V - S - \{v\} \mid v \text{ and } w \text{ are connected by a path in } G[S \cup \{v, w\}]\}$. Note that $R_\pi^G(v) = |Q_G(\pi_{<v}, v)|$, and $|Q_G(S, v)|$ can be computed in $\text{poly}(n)$ time using, for example, depth-first search.

Then define the quantities $\text{TWR}_G(L, S) = \min_{\substack{\pi \in \Pi(V) \\ L \text{ is a prefix of } \pi}} \max_{v \in S} |Q_G(L \cup \pi_{<v}, v)|$ and $\text{TW}_G(S) = \min_{\pi \in \Pi(V)} \max_{v \in S} |Q_G(\pi_{<v}, v)|$. These notations are connected by the relation $\text{TW}_G(G) = \text{TWR}_G(\emptyset, S)$. Note that $\mathbf{tw}(G)$ is equal to $\min_{\pi \in \Pi(V)} \max_{v \in V} R_\pi^G(v) = \text{TW}_G(V)$.

The following lemma gives a way to find optimal fixed bag tree decompositions using the algorithms for finding the optimal linear arrangements:

► **Lemma 10.** *Let $G = (V, E)$ be a graph, and $\chi \subseteq V$ a subset of its vertices. Then $\mathbf{tw}(G, \chi) = \max(\text{TW}_G(V - \chi), |\chi| - 1)$.*

In the final treewidth algorithms, we will also use the following fact:

► **Lemma 11.** *Let $G = (V, E)$ be a graph and $\chi \subseteq V$ a subset of its vertices. Let \mathcal{C} be the set of connected components of $G[V - \chi]$. Then $\mathbf{tw}(G, \chi) = \max_{C \in \mathcal{C}} \mathbf{tw}(G[C \cup \chi], \chi)$.*

Proofs of these lemmas are given in Appendix B.

4.2 Divide & Conquer

The first algorithm is based on the following property:

► **Lemma 12** (Lemma 7 in [6]). *Let $G = (V, E)$ be a graph, $S \subseteq V$, $|S| \geq 2$, $L \subseteq V$, $L \cap S = \emptyset$, $1 \leq k < |S|$. Then*

$$\text{TWR}_G(L, S) = \min_{\substack{S' \subseteq S \\ |S'|=k}} \max(\text{TWR}_G(L, S'), \text{TWR}_G(L \cup S', S - S')).$$

Note that $\text{TWR}_G(L, \{v\}) = |Q_G(L, v)|$ can be calculated in polynomial time. The value we wish to calculate is $\text{TWR}_G(\emptyset, V - \chi)$. Picking $k = |S|/2$ in Lemma 12 and applying Lemma 10, we obtain a $\text{poly}(|V|)4^{|V|-|\chi|}$ deterministic algorithm with polynomial space:

► **Theorem 13** (Theorem 8 in [6]). *Let $G = (V, E)$ be a graph on n vertices and $\chi \subseteq V$ a subset of its vertices. There is an algorithm that calculates $\mathbf{tw}(G, \chi)$ in $O^*(4^{n-|\chi|})$ time and polynomial space.*

Immediately we can prove a quadratic quantum speedup using Grover's search:

► **Theorem 14.** *Let $G = (V, E)$ be a graph on n vertices and $\chi \subseteq V$ a subset of its vertices. There is a bounded-error quantum algorithm that calculates $\mathbf{tw}(G, \chi)$ in $O^*(2^{n-|\chi|})$ time and polynomial space.*

Proof. We can apply quantum minimum finding to check sets S' in Lemma 12, in order to obtain a quadratic speedup over Theorem 13. To avoid the accumulation of error in the recursion, we can use the Grover's search implementation with bounded-error inputs [15]. ◀

4.3 Dynamic programming

The second algorithm is based on the following recurrence:

► **Lemma 15** (Lemma 5 in [6]). *Let $G = (V, E)$ be a graph and $S \subseteq V$, $S \neq \emptyset$. Then*

$$\text{TW}_G(S) = \min_{v \in S} \max(\text{TW}_G(S - \{v\}), |Q_G(S - \{v\}, v)|).$$

Note that in fact Lemma 15 is a special case of Lemma 12 with $L = \emptyset$ and $k = |S| - 1$. This lemma together with Lemma 10 and a dynamic programming technique by Bellman, Held and Karp [4, 14] gives the following algorithm:

► **Theorem 16** (Theorem 6 in [6]). *Let $G = (V, E)$ be a graph on n vertices and $\chi \subseteq V$ a subset of its vertices. There is an algorithm that calculates $\mathbf{tw}(G, \chi)$ in $O^*(2^{n-|\chi|})$ time and space.*

This algorithm calculates the values of $\mathbf{TW}_G(S)$ for all sets S in order of increasing size of the sets, and also stores them all in memory. Such dynamic programming can be sped up quantumly: Ambainis et al. [1] have shown a $O(1.817^n)$ time and space quantum algorithm with QRAM for such problems. Therefore, this gives the following quantum algorithm:

► **Theorem 17.** *Let $G = (V, E)$ be a graph on n vertices and $\chi \subseteq V$ a subset of its vertices. Assuming the QRAM data structure, there is a bounded-error quantum algorithm that calculates $\mathbf{tw}(G, \chi)$ in $O^*(1.816905^{n-|\chi|})$ time and space.*

Note that this algorithm can be used to calculate $\mathbf{TWR}_G(L, S)$. Firstly, $\mathbf{TWR}_G(L, \emptyset) = 0$ and

$$\mathbf{TWR}_G(L, S) = \min_{v \in S} \max(\mathbf{TWR}_G(L, S - \{v\}), |Q_G(L \cup (S - \{v\}), v)|)$$

by Lemma 12. As already mentioned earlier, the value $|Q_G(L \cup (S - \{v\}), v)|$ can be calculated in polynomial time. Hence this recurrence is of the same form as Lemma 15.

► **Theorem 18.** *Let $G = (V, E)$ be a graph on n vertices and $L, S \subseteq V$ be disjoint subsets of vertices. Assuming the QRAM data structure, there is a bounded-error quantum algorithm that calculates $\mathbf{TWR}_G(L, S)$ in $O^*(1.81691^{|S|})$ time and space.*

5 Fomin's and Villanger's algorithm

In this section, we first describe the polynomial space treewidth algorithm of [11]. Afterwards, we summarize the time complexity for the classical algorithm and then for the same algorithm sped up by the quantum tools presented above. The proofs for the theorem in this Section are given in Appendix B.

The algorithm relies on the following, shown implicitly in the proof of Theorem 7.3. of [11].

► **Lemma 19.** *Let $G = (V, E)$ be a graph, and $\beta \in [0, 1]$. There exists an optimal tree decomposition (X, T) of G so that at least one of the following holds:*

- (a) *There exists a bag $\Omega \in X$ such that Ω is a potential maximum clique and there exists a connected component C of $G[V - \Omega]$ such that $|C| \leq \beta n$.*
- (b) *There exists a bag $S \in X$ such that S is a minimal separator and there exist two disjoint connected components C_1, C_2 of $G[V - S]$ such that $N(C_1) = N(C_2) = S$ and $|C_2| \geq |C_1| \geq \beta n$.*

The idea of the algorithm then is to try out all possible potential maximum cliques and minimal separators that conform to the conditions of this lemma, and for each of these sets, to find an optimal tree decomposition of G given that the examined set is a bag of the decomposition using the algorithm from Theorem 13. The treewidth of G then is the minimum width of all examined decompositions.

The potential maximum clique generation is based on the following lemma.

► **Lemma 20** (Lemma 7.1. in [11]). *Let $G = (V, E)$ be a graph. The number of maximum potential cliques Ω of size p such that there exists a connected component C of $G[V - \Omega]$ of size c is at most $n \binom{n-c}{p-1}$.¹ The set of all these cliques can also be generated in time $O^*(\binom{n-c}{p-1})$.*

For the minimal separators, suppose that the size of S is fixed, denote it by s . Note that since C_1 in Lemma 19 is a connected component such that $N(C_1) = S$, then instead of generating minimal separators, we can generate the sets of vertices C with neighborhood size equal to s . The set generated in this way contains in their neighbourhoods all of the minimal separators of size s that we are interested in, and for those sets that do not contain such a separator, the fixed-bag treewidth algorithm will still find some tree decomposition of the graph, albeit not an optimal one. The generation is done using Lemma 7: for a fixed size c of C , the number of such C with exactly s neighbors is at most $n \binom{c+s}{c}$ (the factor of n comes from trying each of n vertices as the fixed vertex $v \in B$). The algorithm generating all such C requires time $O^*(\binom{c+s}{c})$. For a set C , we then find an optimal tree decomposition of G containing $N(C)$ as a fixed bag using the algorithm from Theorem 13. In this way we work through all c from βn to $n - s - |C_2| \leq (1 - \beta)n - s$.

■ **Algorithm 1** The polynomial space algorithm for treewidth.

1. For c from 0 to βn and p from 1 to $n - c$ generate the set of potential maximal cliques Ω of size p with a connected component of $G[V - \Omega]$ of size c using Lemma 20. For each Ω , find $\mathbf{tw}(G, \Omega)$ using Theorem 13.
2. For s from 1 to $(1 - 2\beta)n$ and for c from βn to $(1 - \beta)n - s$ generate the set of subsets C such that $|C| = c$ and $N(C) = S$ using Lemma 7. Let $S = N(C)$; then $\mathbf{tw}(G, S)$ is equal to the maximum of $\mathbf{tw}(G[S \cup C], S)$ and $\mathbf{tw}(G[V - C], S)$ by Lemma 11. Use the algorithm from Theorem 13 to compute these values.
3. Output the minimum width of all examined tree decompositions.

► **Theorem 21** (Theorem 7.3. in [11]). *Algorithm 1 computes the treewidth of a graph with n vertices in $O^*(2.61508^n)$ time and polynomial space.*

5.1 A time-space tradeoff

One might ask whether replacing the $O^*(4^n)$ divide & conquer algorithm from Theorem 13 with the $O^*(2^n)$ dynamic programming algorithm from Theorem 16 in Algorithm 1 can give any interesting complexity. Indeed, we can show the following previously unexamined classical time-space tradeoff.

► **Theorem 4.** *The treewidth of a graph on n vertices can be computed in $O^*(2^n)$ time and $O^*(\sqrt{2^n})$ space.*

We can compare this to the existing treewidth algorithms. The most time-efficient treewidth algorithm runs in time and space $O^*(1.7549^n)$ [11], which is more than $O^*(\sqrt{2^n})$. The polynomial space $O^*(2.6151^n)$ algorithm, of course, is slower than $O^*(2^n)$. The time-space tradeoffs for permutation problems from [16] give $TS \gtrsim 3.93$, where $T \geq 2$ and

¹ The original lemma gives an upper bound if the size of Ω is not fixed, but our statement follows from their proof. We need to fix $|\Omega|$ because in the quantum algorithms, Grover's search will be called for fixed $|\Omega|$ and $|C|$.

$\sqrt{2} \leq S \leq 2$ are the time and space complexities (bases of the exponent to the power of n) of the algorithm. Here, $TS = 2^{\frac{3}{2}} \approx 2.83$, $T = 2$ and $S = \sqrt{2}$. Therefore, Theorem 4 fully subsumes their tradeoff for treewidth. We also note that this tradeoff cannot be “tuned” directly for less time and more space, since the first stage requires $\Theta^*(2^n)$ time for any β .

5.2 Quantum complexity

Now we are ready to examine the quantum versions of the algorithm. First, we consider the analogue of Algorithm 1 sped up quadratically using Grover’s search using Lemma 8 and Theorem 14.

► **Theorem 1.** *There is a bounded-error quantum algorithm that finds the exact treewidth of a graph on n vertices in $O(1.61713^n)$ time and polynomial space.*

Similarly, we can replace the algorithm from Theorem 16 with the quantum dynamic programming algorithm from Theorem 17:

► **Theorem 2.** *Assuming the QRAM data structure, there is a bounded-error quantum algorithm that finds the exact treewidth of a graph on n vertices in $O(1.55374^n)$ time and $O(1.45195^n)$ space.*

6 Improved quantum algorithm

We can see that in Theorem 2 we still have some room for improvement by trading space for time. This can be done using an additional technique. The main idea is to make a global precalculation for $\text{TW}_G(S)$ for all subsets $S \subseteq V$ of size at most αn , for some constant parameter α . Then, as we will see later, these values can be used in all calls of the quantum dynamic programming because of the properties of treewidth. For many such calls, this reduces the $O^*(1.817^d)$ running time to something smaller, which in turn reduces the overall time complexity.

6.1 Asymmetric quantum dynamic programming on the hypercube

We describe our modification to the quantum dynamic programming algorithm by Ambainis et al. [1]. First, we prove the following lemma that allows us to reuse the precalculated DP values on the original graph G in the DP calculation in the subgraphs examined by our algorithms.

► **Lemma 22.** *Let $G = (V, E)$ be a graph, and $\chi \subseteq V$ a subset of its vertices. Suppose that C is a union of some number of connected components of $G[V - \chi]$. Then for any $S \subseteq C$, we have $\text{TW}_{G[C \cup \chi]}(S) = \text{TW}_G(S)$.*

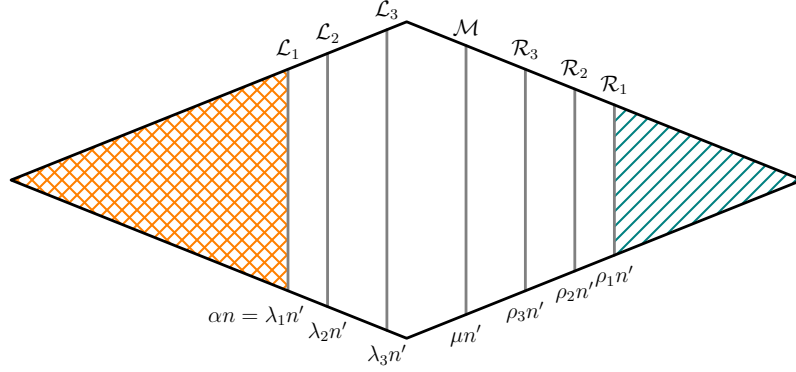
Proof. Examine the permutations π achieving

$$\text{TW}_{G[C \cup \chi]}(S) = \min_{\pi \in \Pi(C \cup \chi)} \max_{v \in S} |Q_G(\pi_{<v}, v)|.$$

As a direct consequence of Lemma 15, there exists such a permutation π with the property that S is its prefix. Now let $\pi' \in \Pi(V)$ be a permutation obtained by adding the vertices of $V - C - \chi$ to π in any order. Examine any vertex $u \in V - C - \chi$ and any $v \in S$. Since u and v are located in different connected components of $G[C - \chi]$, any path from u to v in G passes through some vertex of χ . However, $\pi_{<v} \cap \chi = \emptyset$, as $\pi_{<v} \subset S$. Then we can conclude that $Q_{G[C \cup \chi]}(\pi_{<v}, v) = Q_G(\pi'_{<v}, v)$, as u cannot contribute to Q . Therefore, $\text{TW}_G(S) \leq \text{TW}_{G[C \cup \chi]}(S)$. On the other hand, $\text{TW}_G(S) \geq \text{TW}_{G[C \cup \chi]}(S)$, as additional vertices cannot decrease TW. ◀

11:10 Quantum Speedups for Treewidth

Now we are ready to describe our quantum dynamic programming procedure. Suppose that all values of $\text{TW}_G(S)$ for sets with $|S| \leq \alpha n$ are known and stored in QRAM, where $\alpha \in [0, \frac{1}{2}]$ is some fixed parameter. Suppose that we have fixed a subset $\chi \subseteq V$, and our task is to calculate $\text{tw}(G[C \cup \chi], \chi)$ for a union C of some connected components of $G[V - \chi]$. By Lemma 10, it is equal to $\max(\text{TW}_{G[C \cup \chi]}(C), |\chi| - 1)$. Since $|\chi|$ is known, our goal is to compute $\text{TW}_{G[C \cup \chi]}(C)$.



■ **Figure 1** A schematic representation of layers in the Boolean hypercube with $k = 3$.

Let $n = |V|$ and $n' = |C|$. If $n' \leq \alpha n$, then $\text{TW}_{G[C \cup \chi]}(C) = \text{TW}_G(C)$ by Lemma 22 and is known from the precalculated values. Hence, assume that $n' > \alpha n$. Pick some natural k , we will call this *the number of layers*. Let $\lambda_1 = \frac{\alpha n}{n'}$, and pick constants $\lambda_2 < \dots < \lambda_k < \mu < \rho_k < \dots < \rho_1 < 1$, such that $\lambda_1 < \lambda_2$. Then define collections $L_i = \{S \subseteq C \mid |S| = \lambda_i n'\}$, $\mathcal{M} = \{S \subseteq C \mid |S| = \mu n'\}$ and $\mathcal{R}_i = \{S \subseteq C \mid |S| = \rho_i n'\}$. We call these collections *layers*: we can represent subsets $S \subseteq C$ as vertices on the hypercube of dimension n' ; then these layers are defined as the subsets of vertices with some fixed Hamming weight, see Figure 1. For all sets S corresponding to the vertices in the crosshatched area (such that $|S| \leq \alpha n$), the value of $\text{TW}_{G[C \cup \chi]}(S) = \text{TW}_G(S)$ is known from the assumed precalculation.

Now we will describe the quantum procedure. Denote $G' = G[C \cup \chi]$. Also denote $\text{TW}'_{G'}(S) = \text{TWR}_{G'}(S, C - S)$ and note that $\text{TW}_{G'}(S) = \text{TWR}_{G'}(\emptyset, S)$. Informally, calculating $\text{TW}_{G'}(S)$ means finding the best ordering for the vertices S as a prefix of the permutation, and $\text{TW}'_{G'}(S)$ means finding the best ordering for the vertices $C - S$, where $C - S$ is in the middle of permutation, followed by some ordering of χ .

Algorithm 2 is exactly the algorithm of [1], with the exception that the precalculation is performed only for suffixes (and the precalculation for prefixes comes “for free”). Informally, the idea of the algorithm is to find the optimal path between the vertices s and t with the smallest and highest Hamming weight in the hypercube. First, we use Grover’s search over the vertex v_{k+1} in the middle layer \mathcal{M} . Then we search independently for the best path from s to v_{k+1} and from v_{k+1} to t ; the optimal path from s to t is their concatenation. To find the best path from s to v_{k+1} , we use Grover’s search over the vertex v_k on the layer \mathcal{L}_k such that there exists a path from v_k to v_{k+1} . Then we find the best path from v_k to v_{k+1} by recursively using the $O^*(1.817^{n'})$ algorithm (where n' is the dimension of the hypercube with v_k and v_{k+1} being the smallest and largest weight vertices, respectively). We combine it with the best path from s to v_k , which we find in the similar way (fixing v_{k-1}, \dots, v_1). The value of the optimal path from s to v_1 is known from the global precalculation we assumed

took place before the algorithm. The optimal path from v_{k+1} to t is found analogously; only to know the value of the best path from vertices in \mathcal{R}_1 to t , we have to precalculate these values “from the back” using DP in the beginning of the algorithm.

■ **Algorithm 2** Asymmetric quantum dynamic programming algorithm.

1. For all $S \in \mathcal{R}_1$, calculate and store in QRAM the values $\text{TW}'_{G'}(S)$ using the recurrence

$$\text{TW}'_{G'}(S) = \min_{v \in C-S} \max(\text{TW}'_{G'}(S \cup \{v\}), |Q_{G'}(S, v)|).$$

This follows from Lemma 12 with $k = 1$.

2. Use quantum minimum finding over sets $S \in \mathcal{M}$ to find the answer,

$$\text{TW}_{G'}(C) = \min_{S \in \mathcal{M}} \max(\text{TW}_{G'}(S), \text{TW}'_{G'}(S)).$$

This also follows from Lemma 12 with $k = \mu n'$.

- To find $\text{TW}_{G'}(S)$, we use the recursive procedure $\text{BESTPREFIX}_i(G', S)$. Its value is equal to $\text{TW}_{G'}(S)$, and it requires $S \in \mathcal{L}_i$ (if $i = k + 1$, then $S \in \mathcal{M}$). The needed value is then given by $\text{BESTPREFIX}_{k+1}(G', S)$. The description of $\text{BESTPREFIX}_i(G', S)$:
 - If $i = 1$, return $\text{TW}_{G'}(S) = \text{TW}_G(S)$ that is stored in QRAM.
 - If $1 < i \leq k + 1$, then use quantum minimum finding over the sets $T \in \mathcal{L}_{i-1}$ to find

$$\text{TW}_{G'}(S) = \min_{\substack{T \in \mathcal{L}_{i-1} \\ T \subset S}} \max(\text{BESTPREFIX}_{i-1}(G', T), \text{TWR}_{G'}(T, S - T)).$$

Again, this recurrence follows from Lemma 12 with $k = \lambda_{i-1} n'$. The value of $\text{TWR}_{G'}(T, S - T)$ is calculated by the quantum dynamic programming from Theorem 18 and requires $O^*(1.817^{|S|-|T|})$ time and QRAM space.

- To find $\text{TW}'_{G'}(S)$, we similarly use the recursive procedure $\text{BESTSUFFIX}_i(G', S)$. Its value is equal to $\text{TW}'_{G'}(S)$, and it requires $S \in \mathcal{R}_i$ (if $i = k + 1$, then $S \in \mathcal{M}$). The needed value is then given by $\text{BESTSUFFIX}_{k+1}(G', S)$. The description of $\text{BESTSUFFIX}_i(G', S)$:
 - If $i = 1$, return $\text{TW}'_{G'}(S)$ stored in QRAM from the precalculation in Step 1.
 - If $1 < i \leq k + 1$, then use quantum minimum finding over the sets $T \in \mathcal{R}_{i-1}$ to find

$$\text{TW}'_{G'}(S) = \min_{\substack{T \in \mathcal{R}_{i-1} \\ S \subset T}} \max(\text{TWR}_{G'}(S, T - S), \text{BESTSUFFIX}_{i-1}(G', T)).$$

Again, this recurrence follows from Lemma 12 with $k = \rho_{i-1} n' - \rho_i n'$. The value of $\text{TWR}_{G'}(S, T - S)$ is calculated by the quantum dynamic programming from Theorem 18 and requires $O^*(1.817^{|T|-|S|})$ time and QRAM space.

We will estimate the time complexity of Algorithm 2. The space complexity will not be necessary, because for the final treewidth algorithm it will be dominated by the global precalculation, as we will see later.

11:12 Quantum Speedups for Treewidth

- The time of the precalculation Step 1 is dominated by the size of the layer \mathcal{R}_1 . It is equal to $O^*(|\mathcal{R}_1|) = O^*\binom{n'}{\rho_1 n'}$, which by Lemma 5 is

$$O^*\left(2^{\mathbb{H}(\rho_1)n'}\right).$$

- Let the time of a call of $\text{BESTPREFIX}_i(G', S)$ be T_i , it can be calculated as follows. If $i = 1$,

$$T_1 = O^*(1),$$

as all we need to do is to fetch the corresponding value $\text{TW}_G(S)$ from QRAM. If $i > 1$, then quantum minimum finding examines all $T \in \mathcal{L}_{i-1}$ such that $T \subset S$. The number of such T is $\binom{|S|}{|T|} = \binom{\lambda_i n'}{\lambda_{i-1} n'}$ (for generality, denote $\lambda_{k+1} = \mu$). Again, by Lemma 5, this is at most $2^{\mathbb{H}(\lambda_{i-1}/\lambda_i) \cdot \lambda_i n'}$. The call to $\text{BESTPREFIX}_{i-1}(G', T)$ requires time T_{i-1} and calculating $\text{TWR}_{G'}(T, S - T)$ with the algorithm from Theorem 18 requires time $O^*(1.817^{|S|-|T|}) = O^*(1.817^{(\lambda_i - \lambda_{i-1})n'})$. Putting these estimates together, we get that for $i > 1$,

$$T_i = O^*\left(\sqrt{2^{\mathbb{H}\left(\frac{\lambda_{i-1}}{\lambda_i}\right) \cdot \lambda_i n'}} \cdot \max\left(T_{i-1}, 1.817^{(\lambda_i - \lambda_{i-1})n'}\right)\right).$$

- The time T'_i for $\text{BESTSUFFIX}_i(G', S)$ is calculated analogously. We can check the precalculated values from Step 1 in

$$T'_1 = O^*(1)$$

and (taking $\rho_{k+1} = \mu$) for $i > 1$,

$$T'_i = O^*\left(\sqrt{2^{\mathbb{H}\left(\frac{1-\rho_{i-1}}{1-\rho_i}\right) \cdot (1-\rho_i)n'}} \cdot \max\left(T'_{i-1}, 1.817^{(\rho_{i-1} - \rho_i)n'}\right)\right).$$

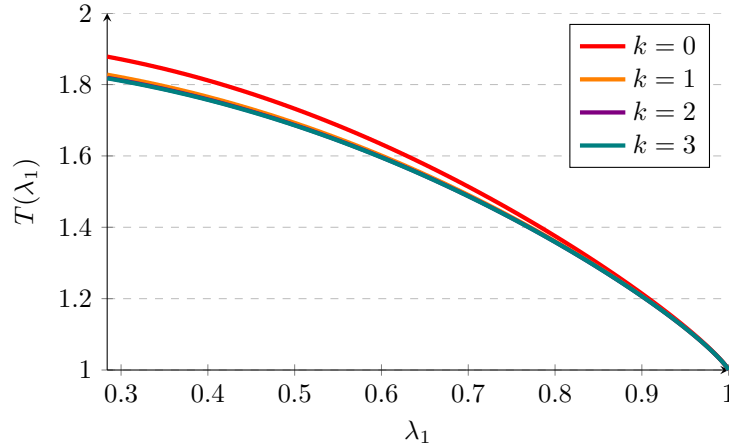
- Lastly, the number of sets examined in the first quantum minimum finding in Step 2 is equal to the size of \mathcal{M} , which is $\binom{n'}{\mu n'} = 2^{\mathbb{H}(\mu)n'}$ by Lemma 5. Therefore, Step 2 requires time

$$O^*\left(\sqrt{2^{\mathbb{H}(\mu)n'}} \cdot \max(T_{k+1}, T'_{k+1})\right).$$

For any of the complexities \mathcal{T} examined here, let's look at $\log_2(\mathcal{T})/n'$; since we are interested in the exponential complexity, we need to investigate only the constant c in $O^*(2^{cn})$. Also note that $\log_2(1.817) \approx 0.862$. This results in the following optimization program

$$\begin{aligned} \text{minimize} \quad & T(\lambda_1) = \max\left(\mathbb{H}(\rho_1), \frac{\mathbb{H}(\mu)}{2} + \max(t_{k+1}, t'_{k+1})\right) \\ \text{s.t.} \quad & \lambda_1 < \dots < \lambda_k < \lambda_{k+1} = \mu = \rho_{k+1} < \rho_k < \dots < \rho_1 < 1 \\ & t_i = \mathbb{H}\left(\frac{\lambda_{i-1}}{\lambda_i}\right) \cdot \lambda_i + \max(t_{i-1}, 0.862(\lambda_i - \lambda_{i-1})) & \forall i \in [2, k+1] \\ & t_1 = 0 \\ & t'_i = \mathbb{H}\left(\frac{1-\rho_{i-1}}{1-\rho_i}\right) \cdot (1-\rho_i) + \max(t'_{i-1}, 0.862(\rho_{i-1} - \rho_i)) & \forall i \in [2, k+1] \\ & t'_1 = 0 \end{aligned}$$

We can solve this program numerically and find the time complexity, depending on the value of λ_1 . Note that for $\lambda_1 \leq 0.28448$ the $O(1.817^{n'})$ symmetric quantum dynamic programming is more efficient, so we don't have to calculate the complexity in that case. Figure 2 shows the time complexity $T(\lambda_1)$ for $k = 0, 1, 2, 3$. We can see that the advantage of adding additional layers quickly becomes negligible.



■ **Figure 2** Running time of the asymmetric quantum dynamic programming algorithm.

Note that with $\lambda_1 \approx 0.28448$, $T(\lambda_1)$ becomes $O^*(1.817^{n'})$, as this is the same parameter for the precalculation layer as in [1]. Thus if it happens that $\alpha n < 0.28448n'$, the asymmetric version of the algorithm will have time complexity larger than $O^*(1.817^{n'})$, so in that case it is better to call the algorithm from Theorem 17. Our procedure for calculating $\text{TW}_{G[C \cup \chi]}(C)$ is given in Algorithm 3.

■ **Algorithm 3** Quantum algorithm calculating $\text{tw}(G[C \cup \chi], \chi)$ assuming global precalculation.

Assume that $\text{TW}_G(S)$ are stored in QRAM for all $|S| \leq \alpha n$.

- If $n' \leq \alpha n$, fetch $\text{TW}_{G[C \cup \chi]}(C) = \text{TW}_G(C)$ from the global precalculation.
- Else if $\alpha n \leq 0.28448n'$, find $\text{TW}_{G[C \cup \chi]}(C)$ using the $O(1.817^{n'})$ algorithm of Theorem 17.
- Else calculate $\text{TW}_{G[C \cup \chi]}(C)$ using Algorithm 2.

Return $\text{tw}(G[C \cup \chi], \chi) = \max(\text{TW}_{G[C \cup \chi]}(C), |\chi| - 1)$.

6.2 Final quantum algorithm

Now we can give the improved quantum dynamic programming algorithm for treewidth, see Algorithm 4. It requires two constant parameters: $\alpha, \beta \in [0, \frac{1}{2}]$. The value αn gives the limit for the global precalculation, and βn is the cutoff point for the two stages as in Algorithm 1.

11:14 Quantum Speedups for Treewidth

■ **Algorithm 4** Improved quantum algorithm for treewidth.

1. Calculate $\text{TW}_G(S)$ for all subsets S such that $|S| \leq \alpha n$ and store them in QRAM.
 2. For c from 0 to βn and p from 1 to $n - c$ examine the set of potential maximal cliques Ω of size p with a connected component of size c . Apply Lemma 8 to Lemma 20 to find the minimum of $\text{tw}(G, \Omega)$ in $O^*\left(\sqrt{\binom{n-c}{p-1}}\right)$ iterations. Calculate the value of $\text{tw}(G, \Omega)$ using Algorithm 3.
 3. For s from 1 to $(1 - 2\beta)n$ and for c from βn to $(1 - \beta)n - s$ examine the set of subsets C such that $|C| = c$ and $|N(C)| = s$. Let $S = N(C)$; then $\text{tw}(G, S)$ is equal to the maximum of $\text{tw}(G[S \cup C], S)$ and $\text{tw}(G[V - C], S)$ by Lemma 11. Find the minimum of $\text{tw}(G, S)$ using Lemma 8 in $O^*\left(\sqrt{\binom{c+s}{s}}\right)$ iterations. Calculate the values of $\text{tw}(G[S \cup C], S)$ and $\text{tw}(G[V - C], S)$ using Algorithm 3.
 4. Return the minimum width of all examined tree decompositions.
-

► **Theorem 3.** *Assuming the QRAM data structure, there is a bounded-error quantum algorithm that finds the exact treewidth of a graph on n vertices in $O(1.53793^n)$ time and space.*

We can calculate the complexity similarly as in Theorem 2.

Proof. First, we choose α such that it balances the time complexity of the global precalculation (Step 1) and the rest of the algorithm (Steps 2–4). The space complexity of this step asymptotically is equal to its time complexity. Therefore, the space complexity of this algorithm is equal to

$$O^*\left(\binom{n}{\alpha n}\right).$$

Denote the time complexity of Algorithm 3 with $|C| = n'$ and some chosen λ_1 by $O^*(\mathcal{T}(\lambda_1)^{n'})$. For fixed α and n' , λ_1 is calculated as $\alpha n/n'$. Then

$$\mathcal{T}(\lambda_1) = \begin{cases} 1, & \text{if } \lambda_1 \geq 1, \\ 1.81691, & \text{if } \lambda_1 \leq 0.28448, \\ T(\lambda_1), & \text{otherwise.} \end{cases}$$

Similarly as we have obtained Equations (1, 2) in the proof of Theorem 21, we can also calculate the time complexity here. The time complexity of Step 2 now is equal to

$$O^*\left(\sum_{c=0}^{\beta n} \sqrt{2^{n-c}} \cdot \mathcal{T}\left(\frac{\alpha n}{c}\right)^c\right).$$

The running time of Step 3 is given by

$$O^*\left(\max_{d=\beta n}^{(1-\beta)n} \sqrt{\binom{n-d}{\beta n}} \cdot \mathcal{T}\left(\frac{\alpha n}{d}\right)^d\right).$$

We can numerically find that $\alpha \approx 0.154468$ and $\beta \approx 0.386401$ balance these complexities, which then are all $O(1.53793^n)$. In our numerical calculation, we have used $k = 3$ for Algorithm 3. ◀

References

- 1 Andris Ambainis, Kaspars Balodis, Jānis Iraids, Martins Kokainis, Krišjānis Prūsis, and Jevgēnijs Vihrovs. Quantum speedups for exponential-time dynamic programming algorithms. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA '19*, pages 1783–1793, USA, 2019. Society for Industrial and Applied Mathematics. doi:10.1137/1.9781611975482.107.
- 2 Stefan Arnborg, Derek G. Corneil, and Andrzej Proskurowski. Complexity of finding embeddings in a k -tree. *SIAM Journal on Algebraic Discrete Methods*, 8(2):277–284, 1987. doi:10.1137/0608024.
- 3 Stefan Arnborg and Andrzej Proskurowski. Linear time algorithms for np-hard problems restricted to partial k -trees. *Discrete Appl. Math.*, 23(1):11–24, April 1989. doi:10.1016/0166-218X(89)90031-0.
- 4 Richard Bellman. Dynamic programming treatment of the travelling salesman problem. *J. ACM*, 9(1):61–63, 1962. doi:10.1145/321105.321111.
- 5 Hans L. Bodlaender. Discovering treewidth. In Peter Vojtáš, Mária Bieliková, Bernadette Charron-Bost, and Ondrej Sýkora, editors, *SOFSEM 2005: Theory and Practice of Computer Science*, pages 1–16, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg. doi:10.1007/978-3-540-30577-4_1.
- 6 Hans L. Bodlaender, Fedor V. Fomin, Arie M. C. A. Koster, Dieter Kratsch, and Dimitrios M. Thilikos. On exact algorithms for treewidth. *ACM Trans. Algorithms*, 9(1), 2012. doi:10.1145/2390176.2390188.
- 7 Christoph Dürr and Peter Høyer. A quantum algorithm for finding the minimum, 1996. arXiv:quant-ph/9607014.
- 8 Uriel Feige, Mohammad Taghi Hajiaghayi, and James R. Lee. Improved approximation algorithms for minimum weight vertex separators. *SIAM Journal on Computing*, 38(2):629–657, 2008. doi:10.1137/05064299X.
- 9 Fedor V. Fomin and Dieter Kratsch. *Exact Exponential Algorithms*. Springer Science & Business Media, 2010.
- 10 Fedor V. Fomin, Daniel Lokshtanov, Saket Saurabh, Michał Pilipczuk, and Marcin Wrochna. Fully polynomial-time parameterized computations for graphs and matrices of low treewidth. *ACM Trans. Algorithms*, 14(3), June 2018. doi:10.1145/3186898.
- 11 Fedor V. Fomin and Yngve Villanger. Treewidth computation and extremal combinatorics. *Combinatorica*, 32:289–308, 2012. doi:10.1007/s00493-012-2536-z.
- 12 Vittorio Giovannetti, Seth Lloyd, and Lorenzo Maccone. Quantum random access memory. *Phys. Rev. Lett.*, 100:160501, 2008. doi:10.1103/PhysRevLett.100.160501.
- 13 Lov K. Grover. A fast quantum mechanical algorithm for database search. In *Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing, STOC '96*, pages 212–219, New York, NY, USA, 1996. Association for Computing Machinery. doi:10.1145/237814.237866.
- 14 Michael Held and Richard M. Karp. A dynamic programming approach to sequencing problems. *Journal of SIAM*, 10(1):196–210, 1962. doi:10.1145/800029.808532.
- 15 Peter Høyer, Michele Mosca, and Ronald de Wolf. Quantum search on bounded-error inputs. In *Automata, Languages and Programming, ICALP'03*, pages 291–299, Berlin, Heidelberg, 2003. Springer-Verlag. doi:10.1007/3-540-45061-0_25.
- 16 Mikko Koivisto and Pekka Parviainen. *A Space-Time Tradeoff for Permutation Problems*, pages 484–492. SODA '10. Society for Industrial and Applied Mathematics, USA, 2010. doi:10.1137/1.9781611973075.41.
- 17 Tuukka Korhonen. A single-exponential time 2-approximation algorithm for treewidth, 2021. arXiv:2104.07463.
- 18 Masayuki Miyamoto, Masakazu Iwamura, Koichi Kise, and François Le Gall. Quantum speedup for the minimum steiner tree problem. In Donghyun Kim, R. N. Uma, Zhipeng Cai, and Dong Hoon Lee, editors, *Computing and Combinatorics*, pages 234–245, Cham, 2020. Springer International Publishing. doi:10.1007/978-3-030-58150-3_19.

- 19 Kazuya Shimizu and Ryuhei Mori. Exponential-time quantum algorithms for graph coloring problems. In Yoshiharu Kohayakawa and Flávio Keidi Miyazawa, editors, *LATIN 2020: Theoretical Informatics*, pages 387–398, Cham, 2020. Springer International Publishing. doi:10.1007/978-3-030-61792-9_31.
- 20 Seiichiro Tani. Quantum Algorithm for Finding the Optimal Variable Ordering for Binary Decision Diagrams. In Susanne Albers, editor, *17th Scandinavian Symposium and Workshops on Algorithm Theory (SWAT 2020)*, volume 162 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 36:1–36:19, Dagstuhl, Germany, 2020. Schloss Dagstuhl–Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.SWAT.2020.36.

A Quantum speedup of the combinatorial lemma

The approach described in Section 3 was formalized by Shimizu and Mori:

► **Lemma 23** (Lemma 4 in [19]). *Let P be a decision problem with parameters n_1, \dots, n_ℓ . Suppose that there is a branching rule $b(P)$ that reduces P to $m_{b(P)}$ problems $P_1, \dots, P_{m_{b(P)}}$ of the same class. Here, P_i has parameters $f_j^{b(P),i}(n_j)$ for $j \in [\ell]$, where $f_j^{b(P),i} \leq n_j$. At least one of the parameters of P_i must be strictly smaller than the corresponding parameter of P . The solution for P is equal to the minimum of the solutions for $P_1, \dots, P_{m_{b(P)}}$.*

Let $U(n_1, \dots, n_\ell)$ be an upper bound on the number of leaves in the computational tree. Assume that the running time of computing $b(P)$, P_i , $f_j^{b(P),i}$ and $U(n_1, \dots, n_\ell)$ is polynomial w.r.t. n_1, \dots, n_ℓ . Suppose that $U(n_1, \dots, n_\ell) \geq \sum_{i=1}^{m_{b(P)}} U(f_1^{b(P),i}(n_1), \dots, f_\ell^{b(P),i}(n_\ell))$. Also suppose that T is the running time for the computation at each of the leaves in the computational tree. Then there is a bounded-error quantum algorithm that computes P and has running time $\text{poly}(n_1, \dots, n_\ell) \sqrt{U(n_1, \dots, n_\ell)} T$.

We apply this to the combinatorial lemma:

Proof of Lemma 7. According to the proof of Lemma 7 in [11], we have that

- $\ell = 2$, $n_1 = b$, $n_2 = f$.
- $b(P)$ splits the problem into $m_{b(P)} = f + b$ problems.
- P_i has parameters $f_1^{b(P),i}(b) = b - 1$ and $f_2^{b(P),i}(f) = f - i + 1$.
- $U(b, f) = \binom{b+f}{f}$.
- $\sum_{i=1}^{m_{b(P)}} U(f_1^{b(P),i}(b), f_2^{b(P),i}(f)) = \sum_{i=1}^{f+b} \binom{f+b-i}{b-1} = \sum_{i=0}^{f+b-1} \binom{f+b-1-i}{b-1} = \binom{b+f}{f} = U(b, f)$.
- Computing $b(P)$, $f_1^{b(P),i}$ and $U(b, f)$ takes time polynomial in b and f ; computing P_i involves contracting two vertices in the graph and can be done in $\text{poly}(n)$ time. ◀

B Proofs of the theorems

To prove Lemma 10, we use the following:

► **Lemma 24** (Lemma 11 in [6]). *Let $C \subseteq V$ induce a clique in a graph $G = (V, E)$. The treewidth of G equals $\max(\text{TW}_G(V - C), |C| - 1)$.*

Proof of Lemma 10. Completing a bag of a tree decomposition into a clique does not change the width of the tree decomposition. The claim then follows from Lemma 24. ◀

Proof of Lemma 11. Let (X, T) be a tree decomposition with the smallest width w that contains χ as a bag. For a connected component $C \in \mathcal{C}$, examine the tree decomposition (X_C, T_C) obtained from (X, T) by removing all vertices not in χ or C from all bags. Clearly,

this is a tree decomposition of $G[C \cup \chi]$ with χ as a bag; as we only have possibly removed some vertices, its width is at most w . Now, examine the tree decomposition obtained by taking all (X_C, T_C) and making χ its common bag. This is a valid tree decomposition, since no two vertices in distinct connected components of \mathcal{C} are connected by an edge. Its width is the maximal width of (X_C, T_C) , therefore at most w . ◀

Proof of Theorem 21. The algorithms from Lemma 7 and Theorem 13 both require polynomial space, hence it holds also for Algorithm 1.

Now we analyze the time complexity; Stage 1 of the algorithm requires time

$$O^* \left(\sum_{c=0}^{\beta n} \sum_{p=1}^{n-c} \binom{n-c}{p-1} 4^{n-p} \right) = O^* \left(\sum_{c=0}^{\beta n} 2^{n-c} 4^c \right) = O^* \left(\max_{c=0}^{\beta n} 2^{n+c} \right) = O^* \left(2^{(1+\beta)n} \right). \quad (1)$$

Stage 2 of the algorithm requires time

$$O^* \left(\sum_{s=1}^{(1-2\beta)n} \sum_{c=\beta n}^{(1-\beta)n-s} \binom{c+s}{c} \max(4^c, 4^{n-c-s}) \right).$$

Note that we can assume that $C = C_1$ and $V - C - S$ contains C_2 (we can check this in polynomial time by finding the connected components of $G[V - S]$); since $|C_2| \geq |C_1|$, we can assume that $n - c - s \geq c$. Hence the complexity becomes

$$O^* \left(\sum_{s=1}^{(1-2\beta)n} \sum_{c=\beta n}^{(1-\beta)n-s} \binom{c+s}{c} 4^{n-c-s} \right) = O^* \left(\max_{s=1}^{(1-2\beta)n} \max_{c=\beta n}^{(1-\beta)n-s} \binom{c+s}{c} 4^{n-c-s} \right).$$

Now denote $d = n - c - s$, then $c + s = n - d$ and we can rewrite the complexity as

$$O^* \left(\max_{d=\beta n}^{(1-\beta)n} \max_{c=\beta n}^{n-d} \binom{n-d}{c} 4^d \right).$$

For any d , the maximum of $\binom{n-d}{c}$ over $c \geq \beta n$ can be one of two cases: if $\beta n \leq \frac{n-d}{2}$, it is equal to $\Theta^*(2^{n-d})$; otherwise it is equal to $\binom{n-d}{\beta n}$. In the first case, for the interval $c \in [\beta n, \frac{n-d}{2}]$, the function being maximized becomes $2^{n-d} 4^d = 2^{n+d}$. Since this function is increasing in d , its maximum is covered by the second case with the smallest c such that $c = \frac{n-d}{2}$ (in case $\beta n \leq \frac{n-d}{2}$). Therefore, the complexity of Stage 2 of the algorithm becomes

$$O^* \left(\max_{d=\beta n}^{(1-\beta)n} \binom{n-d}{\beta n} 4^d \right). \quad (2)$$

Now we are searching for the optimal $\beta \in [0, \frac{1}{2}]$ that balances the complexities (1) and (2). We solve it numerically and obtain $\beta \approx 0.38685$, giving complexity $O^*(2.61508^n)$. ◀

Proof of Theorem 4. First, we look at the time complexity. The time complexity of Stage 1 now is equal to

$$O^* \left(\sum_{c=0}^{\beta n} 2^{n-c} 2^c \right) = O^*(2^n).$$

The time complexity of Stage 2 is equal to

$$O^* \left(\max_{d=\beta n}^{(1-\beta)n} \binom{n-d}{\beta n} 2^d \right) = O^*(2^{n-d} 2^d) = O^*(2^n).$$

11:18 Quantum Speedups for Treewidth

The space complexity of Stage 1 is equal to

$$O^*\left(\max_{c=0}^{\beta n} 2^c\right) = O^*(2^{\beta n}).$$

The space complexity of Stage 2 is equal to

$$O^*\left(\max_{d=\beta n}^{(1-\beta)n} 2^d\right) = O^*(2^{(1-\beta)n}).$$

Therefore, the time complexity of this algorithm is $O^*(2^n)$ and, taking $\beta = \frac{1}{2}$, the space complexity is equal to $O^*(\sqrt{2^n})$. ◀

Proof of Theorem 1. In Algorithm 1, we replace the algorithms from Lemmas 7 and 20 with the quantum algorithm from Lemma 8; the algorithm from Theorem 13 is replaced with the algorithm from Theorem 14. Since all exponential subprocedures now are sped up quadratically, the time complexity becomes

$$O\left(\sqrt{2.61508^n}\right) = O(1.61713^n).$$

The space complexity is still polynomial, as Grover's search additionally uses only polynomial space. ◀

Proof of Theorem 2. The time complexity of the first stage is now equal to

$$O^*\left(\sum_{c=0}^{\beta n} \sqrt{2^{n-c}} \cdot 1.816905^c\right) = O^*\left(\sqrt{2^n} \cdot 1.28475^{\beta n}\right).$$

For the second stage, the time is given by

$$O^*\left(\max_{d=\beta n}^{(1-\beta)n} \sqrt{\binom{n-d}{\beta n}} \cdot 1.816905^d\right).$$

We can numerically find that $\beta \approx 0.3755$ balances these complexities, which then are both $O(1.55374^n)$. The space complexity is

$$O^*\left(1.816905^{\max(\beta n, (1-\beta)n)}\right) = O^*\left(1.816905^{(1-\beta)n}\right) = O(1.45195^n). \quad \blacktriangleleft$$

Qutrit Metaplectic Gates Are a Subset of Clifford+T

Andrew N. Glaudell¹ ✉ 

Booz Allen Hamilton, Atlanta, GA, USA

Department of Mathematics, George Mason University, Fairfax, VA, USA

Neil J. Ross ✉ 

Department of Mathematics and Statistics, Dalhousie University, Halifax, Canada

John van de Wetering ✉ 

Radboud University Nijmegen, The Netherlands

University of Oxford, UK

Lia Yeh ✉ 

Department of Computer Science, University of Oxford, UK

Abstract

A popular universal gate set for quantum computing with qubits is Clifford+ T , as this can be readily implemented on many fault-tolerant architectures. For qutrits, there is an equivalent T gate, that, like its qubit analogue, makes Clifford+ T approximately universal, is injectable by a magic state, and supports magic state distillation. However, it was claimed that a better gate set for qutrits might be Clifford+ R , where $R = \text{diag}(1, 1, -1)$ is the *metaplectic* gate, as certain protocols and gates could more easily be implemented using the R gate than the T gate. In this paper we show that the qutrit Clifford+ R unitaries form a strict subset of the Clifford+ T unitaries when we have at least two qutrits. We do this by finding a direct decomposition of $R \otimes \mathbb{I}$ as a Clifford+ T circuit and proving that the T gate cannot be exactly synthesized in Clifford+ R . This shows that in fact the T gate is more expressive than the R gate. Moreover, we additionally show that it is impossible to find a single-qutrit Clifford+ T decomposition of the R gate, making our result tight.

2012 ACM Subject Classification Theory of computation → Quantum computation theory

Keywords and phrases Quantum computation, qutrits, gate synthesis, metaplectic gate, Clifford+T

Digital Object Identifier 10.4230/LIPIcs.TQC.2022.12

Related Version *Preprint*: <https://arxiv.org/abs/2202.09235>

Funding *Neil J. Ross*: NSERC Discovery Grant.

John van de Wetering: NWO Rubicon Personal Grant.

Lia Yeh: Oxford – Basil Reeve Graduate Scholarship at Oriel College with the Clarendon Fund.

Acknowledgements We would like to thank Alex Christopher Lim and Anikait Mundhra for creating a Python implementation of the constructions in this paper, available at https://github.com/lia-approves/qudit-circuits/tree/main/qutrit_R_from_T.

1 Introduction

Most theoretical work on quantum computing has focussed on *qubits*, two-dimensional quantum systems. However, many proposed physical types of qubits are actually restricted subspaces of higher-dimensional systems, where the natural dimension can be much higher. The restriction to qubits is made for two reasons: the difficulty of precisely controlling quantum systems and the reliance on analogy to classical computers where two-valued bits reign supreme. However, as quantum control continues to improve, researchers have revisited this design choice. In some cases, higher-dimensional *qudits* appear to be the superior option.

¹ Current affiliations of Andrew N. Glaudell: Photonic Inc., Vancouver, BC, Canada; Department of Mathematics, George Mason University, Fairfax, VA, USA; Booz Allen Hamilton, Annapolis Junction, MD, USA



For example, interest in qudit algorithms and physical implementations has risen recently, due to the potential advantages in runtime efficiency, resource requirements, computational space, and noise resilience in communication [23].

For qudits to make a good foundation for a quantum computer, we need methods to achieve fault-tolerance. In qubit-based protocols, one popular paradigm is to rely on the Clifford+ T gate set. This gate set consists of the efficiently simulable Clifford gates that can be implemented directly on many error correcting codes, and the T gate that can be implemented by distilling and injecting magic states [8]. Analogous constructions have been developed for qudits of all dimensions, each one relying on a specific generalization of the Clifford+ T gate set [9].

In this paper we focus on the case of *qutrits*, three-dimensional quantum systems. While qutrits permit the qutrit Clifford+ T gate set, which can be implemented fault-tolerantly on qutrit error correcting codes, analogous to the qubit setting [12], the Clifford+ T gate set is not the only proposed universal fault-tolerant gate set for qutrits. In a series of papers [1, 12, 11, 5, 4, 3, 6] and a patent [7], the non-Clifford qutrit gate of choice is the R gate, also referred to as the FLIP gate, *reflection* gate, or *metaplectic* gate. It is defined as $R := \text{diag}(1, 1, -1)$. This gate was defined in Ref. [1], where it was shown to admit a magic state distillation and injection protocol. As a non-Clifford gate it achieves approximate universality when added to the Clifford gate set [16], as explicitly proved in Ref. [12, Theorem 2]. It can be implemented in a framework of certain weakly-integral non-abelian anyons via braiding and topological measurement [11, 12].

While the definition of the R gate looks very simple, containing only 1's, 0's and a -1 , it is in fact nowhere in the qutrit Clifford hierarchy [10]. This is because for qutrits, Clifford gates are based on the third root of unity $\omega = e^{i2\pi/3}$. Despite this fact, the R gate can still be injected into a qutrit circuit using a repeat-until-success procedure of an R magic state which also allows a distillation protocol [1]. The R gate can hence also be realised fault-tolerantly [1]. Another construction of R is by a measurement-assisted repeat-until-success protocol requiring two ancillary qutrits to probabilistically realise it out of Clifford gates [11]. The R gate has been suggested to be “more powerful in practice” than the T gate [6]. In Ref. [6] they computed the cost of approximating the third level of the Clifford hierarchy in the Clifford+ R (which they refer to as the *metaplectic*) gate set, and claimed that constructing the R gate in the Clifford+ T gate set requires multiple ancillae and repeat-until-success circuits.

In this paper we find evidence in contradiction to these previous assertions. We show that while no single-qutrit Clifford+ T circuit composes to an R gate unitarily², rather unexpectedly the R gate is exactly constructible through a unitary two-qutrit Clifford+ T circuit with T -count 39, which we construct in Section 3. This demonstrates that $R \in \text{Clifford}+T$. Additionally, we prove that the converse is not true, i.e. that $T \notin \text{Clifford}+R$, and hence $\text{Clifford}+R \subsetneq \text{Clifford}+T$. This directly implies any Clifford+ T computation can be exactly implemented through Clifford+ T gates with constant overhead, whereas there exist Clifford+ T circuits whose implementation via Clifford+ R must strictly increase with the desired precision.

This result might seem to contradict the fact that R does not belong anywhere in the Clifford hierarchy, while every Clifford gate and the T gate belongs to the third level C_3 . But recall that while C_1 and C_2 are closed under composition, this is no longer true for the higher levels of the Clifford hierarchy. In particular, it is not true that any circuit built out of Clifford+ T gates is a unitary that belongs to C_3 .

² Unless stated otherwise, we take “single-qutrit” to mean ancilla-free.

The paper is structured as follows. We cover all the basics on qutrit quantum computation and gate synthesis in Section 2. Then in Section 3 we show how to build the R gate as a two-qutrit unitary using only Clifford+ T gates and we prove that it is not possible to do this using just single-qutrit Clifford+ T gates. We finish by demonstrating that T is *not* an element of Clifford+ R so that Clifford+ R is in fact a strict subset of Clifford+ T . We end with some concluding remarks in Section 4.

2 Qutrit Clifford+ T

A qubit is a two-dimensional Hilbert space. Similarly, a qutrit is a three-dimensional Hilbert space. We will write $|0\rangle$, $|1\rangle$, and $|2\rangle$ for the standard computational basis states of a qutrit. Any normalised qutrit state can then be written as

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle + \gamma|2\rangle \quad (1)$$

where $\alpha, \beta, \gamma \in \mathbb{C}$ and $|\alpha|^2 + |\beta|^2 + |\gamma|^2 = 1$.

For a comprehensive overview of quantum computing based on qudits, we refer to the 2020 review by Wang, Hu, Sanders, and Kais [23]. A qudit quantum processor has been experimentally demonstrated on ion trap systems [20] and superconducting circuits [2, 24, 26].

2.1 Pauli gates and permutation gates

Several concepts for qubits extend to qutrits, or more generally to qudits, which are d -dimensional quantum systems. We are concerned with the qudit generalizations of Paulis and Cliffords.

► **Definition 1.** For a d -dimensional qudit, the Pauli X and Z gates are defined as

$$X|k\rangle = |k+1\rangle \quad Z|k\rangle = \omega^k|k\rangle \quad (2)$$

where $\omega := e^{2\pi i/d}$ such that $\omega^d = 1$, and the addition $|k+1\rangle$ is taken modulo d . We define the Pauli group as the set of unitaries generated by compositions and tensor products of the X and Z gates. We write \mathcal{P}_n^d for the Pauli group on n qudits [16, 17].

For qubits this X gate is just the NOT gate while $Z = \text{diag}(1, -1)$. For the duration of this paper we will work solely with qutrits, so we take ω to always be equal to $e^{2\pi i/3}$.

For a qubit there is only one non-trivial permutation of the standard basis states, which is implemented by the X gate. For qutrits there are five non-trivial permutations of the basis states. We call these τ gates and we specify them as τ_L where L is a permutation of the elements $\{0, 1, 2\}$ written in cycle notation. For example, $\tau_{(02)}$ is the permutation which maps $|0\rangle \mapsto |2\rangle$, $|1\rangle \mapsto |1\rangle$, and $|2\rangle \mapsto |0\rangle$. The five non-trivial permutations are then $\tau_{(01)}$, $\tau_{(12)}$, $\tau_{(02)}$, $\tau_{(012)}$, and $\tau_{(021)}$ along with the trivial identity permutation $\mathbb{I} = \tau_{(0)(1)(2)}$. Compositions of these operators are given by $\tau_L \cdot \tau_M = \tau_{L \cdot M}$ with $L \cdot M$ the composition of permutations. Note that $\tau_{(012)} = X$ and $\tau_{(021)} = X^\dagger$.

2.2 Exact synthesis and number rings

One natural question to ask when given a set of gates is to determine which operations can be implemented as a circuit over those gates. This is called the *exact synthesis* problem. One frequently useful notion in addressing exact synthesis is computing the matrix representations of the set of gates in the computational basis and characterizing the *number ring* to which

12:4 Qutrit Metaplectic Gates Are a Subset of Clifford+T

their entries belong. A number ring is a set of numbers which explicitly contains 0 and 1 and is closed under the operations of addition and multiplication. For example, the integers \mathbb{Z} form a number ring.

We can *extend* number rings by considering what happens when we introduce new numbers to the number ring. When we extend the number ring R by α we write $R[\alpha]$ for the ring of formal sums $\sum_j r_j \alpha^j$ where $r_j \in R$. Generally, we extend by an α which is the root of some monic polynomial whose coefficients come from R . If that polynomial has degree p , then all powers of α which are greater than $p - 1$ and appear in an element of $R[\alpha]$ can be reduced via that polynomial. For example, the third root of unity ω solves the monic polynomial $1 + \omega + \omega^2 = 0$ over the integers so that we define $\mathbb{Z}[\omega] = \{a + b\omega \mid a, b \in \mathbb{Z}\}$. Any higher-order powers of ω which might appear in an element of $\mathbb{Z}[\omega]$ can be reduced through its polynomial as for example, $\omega^2 = -1 - \omega$ and $\omega^3 = 1$.

Another common way to modify number rings is to introduce new denominators by *localizing* a ring. For a number ring R we can take any multiplicatively-closed subset μ of R which contains 1 but not 0 and introduce that set of numbers as denominators:

$$\mu^{-1}R := \left\{ \frac{r}{m} \mid r \in R \text{ and } m \in \mu \right\}.$$

► **Definition 2.** *The ring of triadic fractions is the number ring defined by localizing \mathbb{Z} at the set $\mu = \{3^k \mid k \in \mathbb{N}\}$, which we denote as $\mathbb{T} := \mu^{-1}\mathbb{Z} = \{a/3^k \mid a \in \mathbb{Z}, k \in \mathbb{N}\}$.*

The use of number rings to help solve the exact synthesis problem stems from the following statement, attributable to many authors in the field but perhaps most notably to Kliuchnikov, Maslov, and Mosca [18]:

► **Lemma 3.** *Let $\mathcal{G} = \{G_1, \dots, G_k\}$ be a quantum gate set. For all $j \in \{1, \dots, k\}$, let each G_j have the computational basis matrix representation M_j up to a complex global phase such that M_j is a matrix with entries in the number ring R . Then, up to a global phase, the matrix representation of any circuit over \mathcal{G} only has entries in the number ring R .*

It is important to note that Lemma 3 only suffices to *exclude* operations from being representable over a given gate set. To show that a circuit with entries in a particular number ring implies expressibility over a certain gate set is generally equivalent to providing a full solution to the exact synthesis problem.

► **Example 4.** Any qutrit Pauli operation in the computational basis has entries in the number ring $\mathbb{Z}[\omega]$. This follows directly from Lemma 3 and the fact that \mathcal{P}_n^3 is generated by X and Z .

One interesting aspect of $\mathbb{Z}[\omega]$ (and number rings which contain roots of unity in general) is that it contains elements which square to non-square integers. In particular, $(\omega - \omega^2)^2 = (\omega^2 - \omega)^2 = -3$. Note the minus sign here, which is important as $\pm\sqrt{3} \notin \mathbb{Z}[\omega]$. Due to the ubiquity of the Pauli group and the natural appearance of ω , when working with circuits over qutrits it has become increasingly customary to use $\pm(\omega - \omega^2) = \pm i\sqrt{3}$ in place of $\sqrt{3}$ when possible. We make use of this replacement frequently (see, e.g., the Hadamard gate defined below).

2.3 Clifford gates

Another concept that translates to qutrits (or more general qudits) is that of Clifford unitaries.

► **Definition 5.** *Let U be a unitary acting on n qudits. We say that U is Clifford when every Pauli is mapped to another Pauli under conjugation by U . I.e., for any $P \in \mathcal{P}_n^d$ we have $UPU^\dagger \in \mathcal{P}_n^d$.*

Note that the set of n -qudit Cliffords forms a group under composition. For qubits, this group is generated by the S , Hadamard, and CX gates. The same is true for qutrits, for the right generalisation of these gates. To define these it will first be helpful to introduce the notion of qutrit phase gates.

► **Definition 6.** We write $Z(a, b)$ for the phase gate that acts as $Z(a, b)|0\rangle = |0\rangle$, $Z(a, b)|1\rangle = \omega^a|1\rangle$ and $Z(a, b)|2\rangle = \omega^b|2\rangle$ where we take $a, b \in \mathbb{R}$.

We define $Z(a, b)$ in this way, taking a and b to correspond to phases that are multiples of ω , because $Z(a, b)$ will turn out to be Clifford iff a and b are integers. Note that the collection of all $Z(a, b)$ operators constitutes the group of diagonal single-qutrit unitaries modded out by a global phase. Composition of these operations is given by $Z(a, b) \cdot Z(c, d) = Z(a + c, b + d)$.

We will now define the qutrit S gate. For our purposes it will be useful to define it in such a way that it has determinant 1. To do this we will need the ninth-root of unity. Throughout the remainder of the paper, we define $\zeta = e^{2\pi i/9}$.

► **Definition 7.** The qutrit S gate is $S := Z(0, 1)$. I.e., it multiplies the $|2\rangle$ state by ω .

For qubits, the Hadamard interchanges the Z eigenbasis $\{|0\rangle, |1\rangle\}$, and the X basis consisting of the states $|\pm\rangle := \frac{1}{\sqrt{2}}(|0\rangle \pm |1\rangle)$. The same holds for the qutrit Hadamard. In this case the X basis consists of the following states (where we recall from above that $\frac{1}{\omega^2 - \omega} = i/\sqrt{3}$):

$$|+\rangle := \frac{1}{\omega^2 - \omega}(|0\rangle + |1\rangle + |2\rangle) \quad (3)$$

$$|\omega\rangle := \frac{1}{\omega^2 - \omega}(|0\rangle + \omega|1\rangle + \omega^2|2\rangle) \quad (4)$$

$$|\omega^2\rangle := \frac{1}{\omega^2 - \omega}(|0\rangle + \omega^2|1\rangle + \omega|2\rangle) \quad (5)$$

► **Definition 8.** The qutrit Hadamard gate H is the gate that maps $|0\rangle \mapsto |+\rangle$, $|1\rangle \mapsto |\omega\rangle$ and $|2\rangle \mapsto |\omega^2\rangle$. As a matrix:

$$H := \frac{1}{\omega^2 - \omega} \begin{pmatrix} 1 & 1 & 1 \\ 1 & \omega & \omega^2 \\ 1 & \omega^2 & \omega \end{pmatrix} \quad (6)$$

Note that, unlike the qubit Hadamard, the qutrit Hadamard is *not* self-inverse. In fact, we have $H^2 = -\tau_{(12)}$, so that $H^4 = \mathbb{I}$. In particular, $H^\dagger = H^3$. Furthermore, we note that just as the Clifford group in qubits generates certain global phases, the relation $(SH)^3 = -\omega$ implies that global phases of ± 1 , $\pm\omega$, and $\pm\omega^2$ naturally appear in the qutrit Clifford group. The Pauli and S gates we defined all have matrix representations with entries over $\mathbb{Z}[\omega]$. We see that H naturally introduces denominators into our matrices, and so we should localize $\mathbb{Z}[\omega]$ to ensure we can characterize circuits which contain H . Since

$$\frac{\omega^k}{\omega^2 - \omega} = \frac{\omega^k(\omega - \omega^2)}{3}$$

we can introduce the appropriate denominators by localizing at $\mu = \{3^k \mid k \in \mathbb{N}\}$ to get the number ring $\mu^{-1}\mathbb{Z}[\omega]$. Note that this is equivalent to the number ring $\mathbb{T}[\omega]$ which consists of elements $a + b\omega$ where $a, b \in \mathbb{T}$ are triadic fractions.

In Definition 6 we defined the Z phase gate. Similarly, we can define the X phase gates, that give a phase to the X basis states.

12:6 Qutrit Metaplectic Gates Are a Subset of Clifford+T

► **Definition 9.** We define the X phase gates to be $X(a, b) := HZ(a, b)H^\dagger$ where $a, b \in \mathbb{R}$.

We already saw examples of such X phase gates: $X = X(2, 1)$ and $X^\dagger = X(1, 2)$.

Any single-qutrit Clifford can be represented (up to global phase) as a composition of Clifford Z and X phase gates. In particular, we can represent the qutrit Hadamard as follows [15]:

$$H = -Z(2, 2)X(2, 2)Z(2, 2) = -X(2, 2)Z(2, 2)X(2, 2) \quad (7)$$

$$H^\dagger = -Z(1, 1)X(1, 1)Z(1, 1) = -X(1, 1)Z(1, 1)X(1, 1) \quad (8)$$

The final Clifford gate we need is the qutrit CX .

► **Definition 10.** The qutrit CX gate is the two-qutrit gate defined by $CX|i, j\rangle = |i, i + j\rangle$ where the addition is taken modulo 3.

► **Proposition 11.** Let U be a qutrit Clifford unitary. Then up to global phase U can be written as a composition of the S , H and CX gates [16].

From this it easily follows that the $Z(a, b)$ and $X(a, b)$ gates are Clifford if and only if a and b are integers.

► **Corollary 12.** Let U be a qutrit Clifford unitary. Then up to a global phase U has a matrix representation in the computational basis with entries in the number ring $\mathbb{T}[\omega]$.

Proof. This follows from Proposition 11, the definitions of S , H , and CX , and Lemma 3. ◀

2.4 T gates and qutrit controlled gates

Clifford unitaries don't suffice for universal computation, so let's introduce the T gate.

► **Definition 13.** The qutrit T gate is the Z phase gate defined as $T := Z(1/3, -1/3) = \text{diag}(1, \zeta, \zeta^8)$ [19, 9, 17].

Like the qubit T gate, the qutrit T gate belongs to the third level of the Clifford hierarchy, can be injected into a circuit using magic states, and its magic states can be distilled by magic state distillation. This means that we can fault-tolerantly implement this qutrit T gate on many types of quantum error correcting codes. Also as for qubits, the qutrit Clifford+ T gate set is approximately universal, meaning that we can approximate any qutrit unitary using just Clifford gates and the T gate [12, Theorem 1].

The T gate introduces the phase ζ into matrix representations of circuits and thus we should consider extending the previously-defined $\mathbb{T}[\omega]$ by ζ . Note that ζ is a ninth root of unity which solves the cubic polynomial

$$\zeta^3 - \omega = 0 \quad (9)$$

over $\mathbb{T}[\omega]$. In fact, this polynomial has no solutions over $\mathbb{T}[\omega]$, implying that $\zeta \notin \mathbb{T}[\omega]$ (see Appendix A). We thus define the number ring $\mathbb{T}[\zeta]$:

► **Definition 14.** The extension of $\mathbb{T}[\omega]$ by ζ is the number ring $\mathbb{T}[\omega][\zeta] \cong \mathbb{T}[\zeta]$ defined by

$$\mathbb{T}[\omega][\zeta] \cong \mathbb{T}[\zeta] := \{a + b\zeta + c\zeta^2 + d\zeta^3 + e\zeta^4 + f\zeta^5 \mid a, b, c, d, e, f \in \mathbb{T}\}.$$

Any higher powers of ζ that might appear in an expression for an element of $\mathbb{T}[\zeta]$ can be reduced using for instance Eq. (9).

► **Lemma 15.** *Let U be a qutrit Clifford+ T unitary. Then up to a global phase U has a matrix representation in the computational basis with entries in the number ring $\mathbb{T}[\zeta]$.*

Proof. By the definitions of S , H , T , and CX and Lemma 3. ◀

Using T gates, we can construct certain controlled unitaries. When we have an n -qubit unitary U , we can speak of the controlled gate that implements U . This is the $(n + 1)$ -qubit gate that acts as the identity when the first qubit is in the $|0\rangle$ state, and implements U on the last n qubits if the first qubit is in the $|1\rangle$ state.

For qutrits there are however multiple notions of control.

► **Definition 16.** *Let U be a qutrit unitary. Then the $|2\rangle$ -controlled U is the unitary $|2\rangle$ - U that acts as*

$$|0\rangle \otimes |\psi\rangle \mapsto |0\rangle \otimes |\psi\rangle \quad |1\rangle \otimes |\psi\rangle \mapsto |1\rangle \otimes |\psi\rangle \quad |2\rangle \otimes |\psi\rangle \mapsto |2\rangle \otimes U|\psi\rangle$$

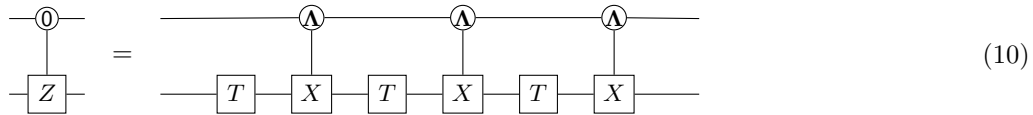
I.e., it implements U on the last qutrits if and only if the first qutrit is in the $|2\rangle$ state.

Note that by conjugating the first qutrit with X or X^\dagger gates we can make the gate also be controlled on the $|1\rangle$ or $|0\rangle$ state.

A different notion of qutrit control was introduced by Bocharov, Roetteler, and Svore [6]: Given a qutrit unitary U they define $\Lambda(U) |c\rangle |t\rangle = |c\rangle \otimes (U^c |t\rangle)$. I.e., apply the unitary U a number of times equal to the value of the control qutrit, so that if the control qutrit is $|2\rangle$ we apply U^2 to the target qutrits. Note that we can get this notion of control from the former one: just apply a $|1\rangle$ -controlled U , followed by a $|2\rangle$ -controlled U^2 . The Clifford CX gate defined earlier is in this notation equal to $\Lambda(X)$.

Adding controls to a Clifford gate generally makes it non-Clifford. In the case of the CX gate, which is $\Lambda(X)$, it is still Clifford, but the $|2\rangle$ -controlled X is *not*.

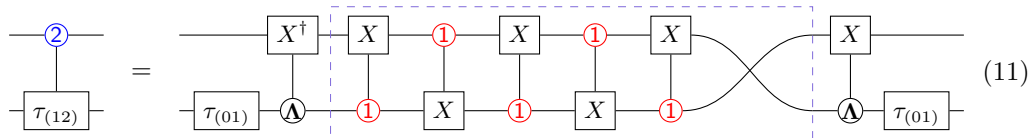
As shown by Bocharov, Roetteler, and Svore, the $|0\rangle$ -controlled Z gate can be constructed by the following 3 T gate circuit [6, Figure 6]:



By conjugating the control qutrit by either X^\dagger or X , the $|1\rangle$ - and $|2\rangle$ -controlled versions are respectively obtained. Taking the adjoint of Eq. (10) has the effect of changing the target operation from Z to Z^\dagger . Finally, note that we can also use this construction for controlled X and X^\dagger gates by conjugating the target qutrit of Eq. (10) by an H or H^\dagger gate. By adapting a different circuit from Ref. [6] we can also construct the other controlled X permutation gates.

► **Lemma 17.** *The $|2\rangle$ -controlled versions of the $\tau_{(01)}$, $\tau_{(02)}$, and $\tau_{(12)}$ gates can be implemented unitarily using Clifford+ T gates without ancillae, with a T -count of 15.*

Proof. The $|2\rangle$ -controlled $\tau_{(12)}$ gate can be constructed as follows:



The dashed box shows the Clifford equivalent gate given by Bocharov, Roetteler, and Svore [6, 4] upon which the construction is based. The $|2\rangle$ -controlled $\tau_{(01)}$ and $|2\rangle$ -controlled $\tau_{(02)}$ gates follow from Clifford equivalence. ◀

Note that the blue and red color of the controls here is just to visually indicate more clearly which type of control is meant. The colors have no further significance.

3 Results

Previous implementations of the R gate require either distillation [1] or probabilistic creation of the $\text{diag}(1, 1, -1)$ state [11, 6]; both approaches then necessitate injection by a repeat-until-success protocol. Here we present a new approach, which implements R unitarily over the qutrit Clifford+ T gate set. As we will discuss later, it is actually impossible to exactly build the R gate from only single-qutrit Clifford+ T gates. However, we *can* construct the two-qutrit $R \otimes \mathbb{I}$ unitarily using Clifford+ T gates. We will do this³ by showing how to construct certain $|2\rangle$ -controlled gates and then using the following observation:

$$\begin{array}{c} \textcircled{2} \quad \textcircled{2} \quad \textcircled{2} \\ | \quad | \quad | \\ \boxed{-H} \quad \boxed{-H} \quad \boxed{-H^2} \end{array} = \begin{array}{c} \textcircled{2} \\ | \\ \boxed{-\mathbb{I}} \end{array} = \begin{array}{c} \textcircled{2} \\ | \\ \boxed{R} \end{array} \quad (12)$$

This works because $H^4 = \mathbb{I}$, and the fact that global phases become *local* phases when adding control wires to them. Here we have a *controlled* global phase because the global phase of -1 is applied to the target if and only if the control is in the $|2\rangle$ state. Therefore, this is an instance of phase kickback: The action of the $|2\rangle$ -controlled $-\mathbb{I}$ gate is identical to applying $R \otimes \mathbb{I}$, i.e. the R gate to the control qutrit and identity on the target.

The $|2\rangle$ -controlled $-H^2 = \tau_{(12)}$ was constructed as a Clifford+ T circuit in Eq. (11). It hence remains to show how we can construct the $|2\rangle$ -controlled $-H$ gate in Clifford+ T . We can build this using the $|2\rangle$ -controlled S gate. Note that for our purposes here our constructions are up to a controlled global phase, arising from our particular choice of global phase convention to define the Clifford gates in relation to a number ring. In forthcoming work [25], we chose a different convention enabling us do away with the controlled global phases here and exactly construct all multiple-controlled Clifford+ T gates in Clifford+ T .

► **Lemma 18.** *The $|2\rangle$ -controlled S gate can be constructed unitarily without ancillae, up to a controlled global phase of ζ^8 , using only Clifford+ T gates, with T -count 8.*

Proof. The correctness can be verified by direct computation of the following circuit.

$$\begin{array}{c} \textcircled{2} \\ | \\ \boxed{\zeta^8 S} \end{array} = \begin{array}{c} \textcircled{2} \quad \textcircled{2} \\ | \quad | \\ \tau_{(01)} \quad T^\dagger \quad \tau_{(01)} \quad X \quad \tau_{(01)} \quad T \quad \tau_{(01)} \quad X^\dagger \end{array} \quad (13)$$

Alternatively, it is easy to see that this circuit does nothing if the first qutrit is in the $|0\rangle$ or $|1\rangle$ state, as the τ_{01} gates cancel, so that the T can cancel with the T^\dagger . Otherwise, if the first qutrit is $|2\rangle$, then the middle permutations combine to $\tau_{(01)}X\tau_{(01)} = \tau_{(012)} = X^\dagger$. When the T is pushed through this, the phases get permuted, and when combined with the T^\dagger gives an S gate up to global phase. ◀

► **Corollary 19.** *The $|2\rangle$ -controlled $Z(2, 2) = \text{diag}(1, \omega^2, \omega^2)$ gate can be constructed unitarily without ancillae, up to a controlled global phase of ζ^2 , using only Clifford+ T gates, with T -count 8.*

Proof. Use the following circuit:

$$\begin{array}{c} \textcircled{2} \\ | \\ \boxed{\zeta^2 Z(2, 2)} \end{array} = \begin{array}{c} \textcircled{2} \\ | \\ \tau_{(02)} \quad \boxed{\zeta^8 S} \quad \tau_{(02)} \end{array} \quad (14)$$

Its correctness can be verified by direct computation, or by commuting S and $\tau_{(02)}$. ◀

³ Implemented at https://github.com/lia-approves/qudit-circuits/tree/main/qutrit_R_from_T.

► **Lemma 20.** *The $|2\rangle$ -controlled H gate can be constructed unitarily without ancillae, up to a controlled global phase of -1 , using Clifford+ T gates with T -count 24.*

Proof. In the construction given below, we use the decomposition of H into alternating Z and X Clifford rotations of Eq. (7).

$$\begin{array}{c}
 \text{---} \text{---} \text{---} \\
 \text{---} \text{---} \text{---} \\
 \text{---} \text{---} \text{---} \\
 \text{---} \text{---} \text{---}
 \end{array}
 \quad = \quad
 \begin{array}{c}
 \text{---} \text{---} \text{---} \\
 \text{---} \text{---} \text{---} \\
 \text{---} \text{---} \text{---} \\
 \text{---} \text{---} \text{---}
 \end{array}
 \quad (15)$$

To construct the controlled H up to a controlled global phase, we apply Eq. (14), conjugating the target by Hadamards for the X rotations per Definition 9. As we require three such gates, their combined $|2\rangle$ -controlled global phase becomes $\zeta^{2 \cdot 3} = \zeta^6$. As $\zeta^9 = 1$, the necessary correction is to apply the $|2\rangle$ -controlled global phase of ζ^3 gate, i.e. the $Z(0, 1) \otimes \mathbb{I}$ gate. ◀

We can now construct the R gate in Clifford+ T . However, direct substitution of the 24 T -count $|2\rangle$ -controlled H gate of Lemma 20 into Eq. (12) yields a T -count 63 construction. We can do better by combining the two iterations of the controlled H in a smarter way.

► **Theorem 21.** *The qutrit R gate can be constructed unitarily in Clifford+ T with T -count 39, provided there is a borrowed (i.e. returned to its starting state) ancilla available.*

Proof. The equality of the circuits below can be verified by direct computation or by noting that it applies $Z(3, 3) = \mathbb{I}$ to the target when the control is $|0\rangle$ or $|1\rangle$, and $H^2 = -\tau_{(12)}$ otherwise.

$$\begin{array}{c}
 \text{---} \text{---} \text{---} \\
 \text{---} \text{---} \text{---} \\
 \text{---} \text{---} \text{---} \\
 \text{---} \text{---} \text{---}
 \end{array}
 \quad = \quad
 \begin{array}{c}
 \text{---} \text{---} \text{---} \\
 \text{---} \text{---} \text{---} \\
 \text{---} \text{---} \text{---} \\
 \text{---} \text{---} \text{---}
 \end{array}
 \quad (16)$$

To get a circuit for the R gate we simply bring the $|2\rangle$ -controlled $\tau_{(12)}$ to the other side (as it is its own inverse). The total T -count of the resulting circuit is then $15 + 3 \cdot 8 = 39$. ◀

As we can construct the R gate as a Clifford+ T circuit, any unitary that can be exactly constructed in the Clifford+ R gate set can then be exactly (as opposed to approximately) constructed in the Clifford+ T gate set. Although do note that our conversion presently seems rather inefficient, as the circuit in Eq. (12) requires 39 T gates.

► **Corollary 22.** *The Clifford+ R gate set is a subset of the Clifford+ T gate set.*

A natural question to ask now is whether we can do better. Do we really need two qutrits to write the R gate as a Clifford+ T unitary? The answer is *yes*: it is not possible to construct the R gate using just single-qutrit Clifford+ T gates. This follows from the normal form that was found for single-qutrit Clifford+ T unitaries in Ref. [14]. Since the proof of this is rather technical we present the details in Appendix B, and just give a sketch here.

The group of 3×3 unitary matrices acts on the 8-dimensional real vector space of traceless Hermitian matrices. This action defines, for each 3×3 unitary matrix U , an 8×8 real matrix \bar{U} known as the *adjoint representation* of U . One can then gather information about U by studying its adjoint representation \bar{U} . In particular, it is a consequence of the normal forms for single-qutrit Clifford+ T circuits introduced in Ref. [14] that the adjoint

12:10 Qutrit Metaplectic Gates Are a Subset of Clifford+T

representation of a single-qutrit Clifford+ T operator has a very specific block matrix form (see Proposition 33 in Appendix B below). It can then be shown by computation that \overline{R} is not of the appropriate form and therefore not Clifford+ T .

Another natural question is the converse to Corollary 22: is the T gate included in Clifford+ R ? I.e., is the inclusion of Clifford+ R within Clifford+ T strict? We will show that this is indeed the case. We begin by considering matrix representations of circuits over Clifford+ R .

► **Lemma 23.** *Let U be a qutrit Clifford+ R unitary. Then up to a global phase U has a matrix representation in the computational basis with entries in the number ring $\mathbb{T}[\omega]$.*

Proof. By the definitions of S , H , R , and CX and Lemma 3. ◀

► **Proposition 24.** $T \notin \text{Clifford}+R$

Proof. We have $T \in \text{Clifford}+R$ if there exists a unitary circuit over Clifford+ R which performs the operation $T \otimes \mathbb{I}_n$ up to a global phase for some $n \in \mathbb{N}$ where \mathbb{I}_n is the n -qutrit identity. In the computational basis, $T \otimes \mathbb{I}$ has a matrix representation with entries from the set $\{0, 1, \zeta, \zeta^8\}$. By Lemma 23, we know that if $T \otimes \mathbb{I}$ permits an exact circuit over Clifford+ R we must have $\{0, c, c\zeta, c\zeta^8\} \subset \mathbb{T}[\omega]$ for at least some global phase $c \in \mathbb{C}$ which satisfies $c^*c = 1$. As $\mathbb{T}[\omega]$ is closed under conjugation, we then also have $c^* \in \mathbb{T}[\omega]$, and as it is closed under multiplication we then have $c^*c\zeta = \zeta \in \mathbb{T}[\omega]$. However, it is well-known that $\zeta \notin \mathbb{T}[\omega]$, and so there exists no such global phase c (see Appendix A). Hence, no such suitable c exists. As n was arbitrary, we conclude that no Clifford+ R circuit exactly implements T in the computational basis. ◀

► **Corollary 25.** $\text{Clifford}+R \subsetneq \text{Clifford}+T$.

4 Conclusion

In summary, we showed that the universal fault-tolerant qutrit Clifford+ R gate set is a subset of Clifford+ T , by providing a two-qutrit, T -count 39 unitary Clifford+ T construction of the R gate. We prove that our construction is optimal in the number of qutrits by using the single-qutrit Clifford+ T normal form of Glaudell, Ross, and Taylor [14] to show there is no single-qutrit construction. Moreover, we prove that Clifford+ R is a *strict* subset of Clifford+ T by showing that regardless of the number of ancillae qutrits, the T gate is impossible to exactly synthesize unitarily in the Clifford+ R gate set.

This result is surprising for several reasons. While a number of papers have studied the Clifford+ R gate set, it was not known that it is a subset of Clifford+ T , much less a strict subset. Therefore, in contrast to what was previously believed, it looks like the T gate could be more powerful in practice than the R gate. In fact, we find that Clifford+ T is strictly more powerful (at least asymptotically and for exact synthesis) than Clifford+ R as Clifford+ T can exactly synthesize every gate in Clifford+ R up to a constant factor of overhead, while the converse is not true. We have reason to believe that the additional gates Clifford+ T can exactly represent are important in practice. In Ref. [4] they conjectured that not all ternary classical reversible gates can be exactly represented in Clifford+ R , while we have shown in follow-up work that they can all be efficiently constructed in Clifford+ T [25]. Further analysis is required to better understand the implications of our results with regards to qutrit algorithms in practice, building upon the comparison between these two gate sets for Shor's algorithm in Ref. [6]. Let us note that using our construction, much of the work

done on Clifford+ R can now be directly translated to the Clifford+ T setting. For example, the universal approximate synthesis algorithms of [5, 3] can now also be used to synthesise Clifford+ T circuits.

Our results demonstrate a way in which qutrit Clifford+ T is different from that of qubit Clifford+ T . While all the one-qubit Clifford+ T circuits that can be constructed with and without ancillae coincide [18], our result shows that this is not true for qutrits, as the single-qutrit R gate cannot be constructed in single-qutrit Clifford+ T , but can be constructed using one borrowed ancilla.

A natural starting point for future work is to find a lower T -count decomposition of the R gate, as it seems unlikely that the best possible construction would require 39 T gates. It might be possible to find a lower bound on the necessary T -count to prepare the R state by using the resource theory of non-stabiliser states, for instance the mana [21] and thauma [22] measures of magic. Alternatively, there might also be a normal form for multi-qutrit Clifford+ T unitaries which is T -optimal, which would then also give us an optimal decomposition of the R gate.

Finally, our results pave the way to deriving a full characterisation of which qutrit unitaries can be exactly implemented over the Clifford+ T gate set. We conjecture that, as in the qubit case [13], any qutrit unitary with entries in $\mathbb{T}[\zeta]$ can be exactly synthesised over Clifford+ T .

References

- 1 Hussain Anwar, Earl T Campbell, and Dan E Browne. Qutrit magic state distillation. *New Journal of Physics*, 14(6):063006, June 2012. doi:10.1088/1367-2630/14/6/063006.
- 2 M. S. Blok, V. V. Ramasesh, T. Schuster, K. O'Brien, J. M. Kreikebaum, D. Dahlen, A. Morvan, B. Yoshida, N. Y. Yao, and I. Siddiqi. Quantum information scrambling on a superconducting qutrit processor. *Phys. Rev. X*, 11:021010, April 2021. doi:10.1103/PhysRevX.11.021010.
- 3 Alex Bocharov. A note on optimality of quantum circuits over metaplectic basis. *Quantum Information and Computation*, 18, June 2016. doi:10.26421/QIC18.1-2-1.
- 4 Alex Bocharov, Shawn Cui, Martin Roetteler, and Krysta Svore. Improved quantum ternary arithmetics. *Quantum Information and Computation*, 16:862–884, July 2016. doi:10.26421/QIC16.9-10-8.
- 5 Alex Bocharov, Xingshan Cui, Vadym Kliuchnikov, and Zhenghan Wang. Efficient topological compilation for a weakly integral anyonic model. *Physical Review A*, 93(1), January 2016. doi:10.1103/physreva.93.012313.
- 6 Alex Bocharov, Martin Roetteler, and Krysta M. Svore. Factoring with qutrits: Shor's algorithm on ternary and metaplectic quantum architectures. *Phys. Rev. A*, 96:012306, July 2017. doi:10.1103/PhysRevA.96.012306.
- 7 Alexei Bocharov, Zhenghan Wang, Xingshan Cui, and Vadym Kliuchnikov. Efficient topological compilation for metaplectic anyon model, May 2021. United States Patent and Trademark Office. Assignee: Microsoft Technology Licensing, LLC.
- 8 Sergey Bravyi and Alexei Kitaev. Universal quantum computation with ideal Clifford gates and noisy ancillas. *Phys. Rev. A*, 71:022316, February 2005. doi:10.1103/PhysRevA.71.022316.
- 9 Earl T. Campbell, Hussain Anwar, and Dan E. Browne. Magic-state distillation in all prime dimensions using quantum Reed-Muller codes. *Phys. Rev. X*, 2:041021, December 2012. doi:10.1103/PhysRevX.2.041021.
- 10 Shawn X. Cui, Daniel Gottesman, and Anirudh Krishna. Diagonal gates in the Clifford hierarchy. *Phys. Rev. A*, 95:012329, January 2017. doi:10.1103/PhysRevA.95.012329.
- 11 Shawn X. Cui, Seung-Moon Hong, and Zhenghan Wang. Universal quantum computation with weakly integral anyons. *Quantum Information Processing*, 14(8):2687–2727, May 2015. doi:10.1007/s11128-015-1016-y.

- 12 Shawn X. Cui and Zhengnan Wang. Universal quantum computation with metaplectic anyons. *Journal of Mathematical Physics*, 56(3):032202, March 2015. doi:10.1063/1.4914941.
- 13 Brett Giles and Peter Selinger. Exact synthesis of multiqubit Clifford+T circuits. *Physical Review A*, 87(3), March 2013. doi:10.1103/physreva.87.032332.
- 14 Andrew N. Glaudell, Neil J. Ross, and Jacob M. Taylor. Canonical forms for single-qutrit Clifford+T operators. *Annals of Physics*, 406:54–70, July 2019. doi:10.1016/j.aop.2019.04.001.
- 15 Xiaoyan Gong and Quanlong Wang. Equivalence of local complementation and Euler decomposition in the qutrit zx-calculus, 2017. arXiv:1704.05955.
- 16 Daniel Gottesman. Fault-tolerant quantum computation with higher-dimensional systems. *Chaos, Solitons & Fractals*, 10(10):1749–1758, September 1999. doi:10.1016/s0960-0779(98)00218-5.
- 17 Mark Howard and Jiri Vala. Qudit versions of the qubit $\pi/8$ gate. *Phys. Rev. A*, 86:022316, August 2012. doi:10.1103/PhysRevA.86.022316.
- 18 Vadym Kliuchnikov, Dmitri Maslov, and Michele Mosca. Fast and efficient exact synthesis of single-qubit unitaries generated by Clifford and T gates. *Quantum Info. Comput.*, 13(7–8):607–630, July 2013.
- 19 Shiroman Prakash, Akalank Jain, Bhakti Kapur, and Shubangi Seth. Normal form for single-qutrit Clifford+T operators and synthesis of single-qutrit gates. *Physical Review A*, 98(3), September 2018. doi:10.1103/physreva.98.032304.
- 20 Martin Ringbauer, Michael Meth, Lukas Postler, Roman Stricker, Rainer Blatt, Philipp Schindler, and Thomas Monz. A universal qudit quantum processor with trapped ions, 2021. arXiv:2109.06903.
- 21 Victor Veitch, S A Hamed Mousavian, Daniel Gottesman, and Joseph Emerson. The resource theory of stabilizer quantum computation. *New Journal of Physics*, 16(1):013009, January 2014. doi:10.1088/1367-2630/16/1/013009.
- 22 Xin Wang, Mark M. Wilde, and Yuan Su. Efficiently computable bounds for magic state distillation. *Physical Review Letters*, 124(9), March 2020. doi:10.1103/physrevlett.124.090505.
- 23 Yuchen Wang, Zixuan Hu, Barry C. Sanders, and Sabre Kais. Qudits and high-dimensional quantum computing. *Frontiers in Physics*, 8:479, 2020. doi:10.3389/fphy.2020.589504.
- 24 Biaoliang Ye, Zhen-Fei Zheng, Yu Zhang, and Chui-Ping Yang. Circuit QED: single-step realization of a multiqubit controlled phase gate with one microwave photonic qubit simultaneously controlling $n - 1$ microwave photonic qubits. *Optics Express*, 26(23):30689, November 2018. doi:10.1364/oe.26.030689.
- 25 Lia Yeh and John van de Wetering. Constructing all qutrit controlled Clifford+T gates in Clifford+T. In *International Conference on Reversible Computation*. Springer, In press. arXiv:2204.00552.
- 26 M. A. Yurtalan, J. Shi, M. Kononenko, A. Lupascu, and S. Ashhab. Implementation of a Walsh-Hadamard gate in a superconducting qutrit. *Phys. Rev. Lett.*, 125:180504, October 2020. doi:10.1103/PhysRevLett.125.180504.

A $\zeta \notin \mathbb{T}[\omega]$

We provide an elementary proof that primitive ninth roots of unity are not elements of $\mathbb{T}[\omega]$. Note that $\zeta \in \mathbb{T}[\omega]$ only if $\zeta \in \mathbb{Q}[\omega]$, where $\mathbb{Q}[\omega]$ is a field. As $\{1, \omega\}$ forms a basis for $\mathbb{Q}[\omega]$ over \mathbb{Q} , if $\zeta \in \mathbb{Q}[\omega]$ we would necessarily require some $a, b \in \mathbb{Q}$ such that

$$\zeta = a + b\omega \implies \zeta^3 = (a + b\omega)^3 = \omega.$$

Expanding, reducing powers using $1 + \omega + \omega^2 = 0$, and collecting terms, we find

$$(a^3 - 3ab^2 + b^3) + (3a^2b - 3ab^2)\omega = \omega.$$

Therefore, by equating coefficients of our basis elements on each side we conclude that we need

$$a^3 - 3ab^2 + b^3 = 0 \quad (17)$$

$$3a^2b - 3ab^2 = 1. \quad (18)$$

Note that clearly $a, b \neq 0$ if Eq. 18 is to be satisfied. Letting $r = a/b \in \mathbb{Q}$, we rearrange Eq. 17 and find

$$r^3 - 3r + 1 = 0. \quad (19)$$

Since $r \neq 0$, let $r = s/t$ for $s, t \in \mathbb{Z}$, $s, t \neq 0$, and $\gcd(s, t) = 1$ without loss of generality. Necessarily, we would have

$$s^3 - 3st^2 + t^3 = 0. \quad (20)$$

For any prime $p \mid s$, we clearly have $p \mid t^3$ implying $p \mid t$. Similarly, for any prime $q \mid t$, we must have $q \mid s^3$ and thus $q \mid s$. As we have assumed $\gcd(s, t) = 1$, we conclude that no prime can divide s nor t and so s, t must be units in \mathbb{Z} as both are necessarily nonzero. No combination of $s, t = \pm 1$ satisfies Eq. 20, and thus we conclude no $r \in \mathbb{Q}$ satisfies Eq. 19. From this, we deduce there are no $a, b \in \mathbb{Q}$ such that

$$\zeta = a + b\omega$$

and thus $\zeta \notin \mathbb{Q}[\omega] \implies \zeta \notin \mathbb{T}[\omega]$.

B The R gate is not a single-qutrit Clifford+ T unitary

We start with a set of definitions. These are based on the work done in Ref. [14].

► **Definition 26.** Let $K = \{2^k \mid k \in \mathbb{N}\}$, $\alpha = \sin(2\pi/9)$, and $L = \{\alpha^k \mid k \in \mathbb{N}\}$. We define the following number rings:

$$\begin{aligned} \mathbb{D} &:= K^{-1}\mathbb{Z} = \left\{ \frac{a}{2^k} \mid a \in \mathbb{Z} \text{ and } k \in \mathbb{N} \right\} \\ \mathbb{D}[\alpha] &= \{a + b\alpha + c\alpha^2 + d\alpha^3 + e\alpha^4 + f\alpha^5 \mid a, b, c, d, e, f \in \mathbb{D}\} \\ \mathbb{A} &:= L^{-1}\mathbb{D}[\alpha] = \left\{ \frac{a}{\alpha^k} \mid a \in \mathbb{D}[\alpha] \text{ and } k \in \mathbb{N} \right\} \end{aligned}$$

Additionally, we will rely on the following quotient ring:

► **Definition 27.** Let $\mathbb{Z}_3 := \mathbb{Z}/(3)$ be the ring of integers modulo 3.

Using our definitions we can introduce the following ring homomorphism:

► **Definition 28.** Let $\rho : \mathbb{D}[\alpha] \rightarrow \mathbb{Z}_3$ be the ring homomorphism defined by $\rho(q) = q \pmod{\alpha}$ for $q \in \mathbb{D}[\alpha]$. In particular, $\rho(1/2) = 2$, $\rho(3) = 0$, and $\rho(\alpha) = 0$.

To account for powers of α that appear in the denominator of elements of \mathbb{A} , we also introduce the following terminology:

► **Definition 29.** Let $q \in \mathbb{A}$. There always exists some $k \in \mathbb{N}$ for which $\alpha^k q \in \mathbb{D}[\alpha]$. We call k a denominator exponent of q , and the least such k is called the least denominator exponent (LDE). The LDE of a vector or matrix over \mathbb{A} is defined as the largest LDE of their individual elements.

12:14 Qutrit Metaplectic Gates Are a Subset of Clifford+T

► **Definition 30.** Let $q \in \mathbb{A}$ and let k be a denominator exponent of q . Then the k -residue of q , $\rho_k(q)$ is defined as

$$\rho_k(q) := \rho(\alpha^k q) \in \mathbb{Z}_3.$$

The k -residue of a vector or matrix is defined component-wise.

The rings we introduced will encompass the entries of Clifford+T matrices in a certain representation called the adjoint representation, which we can describe as follows. Consider the space \mathbb{H} of traceless 3×3 Hermitian matrices. This space forms an 8-dimensional real vector space and can be endowed with an inner product by defining $\langle M, M' \rangle = \text{Tr}(M^\dagger M')$, for any $M, M' \in \mathbb{H}$. As the trace is both cyclic and fixed under transposition of arguments, we have $\langle M, M' \rangle^* = \langle M, M' \rangle$ so that inner product of two traceless Hermitian matrices is necessarily real. It is straightforward to verify that if U is a 3×3 unitary matrix, then conjugation by U defines a linear operator on \mathbb{H} .

► **Definition 31.** Let U be a 3×3 unitary matrix. We define the linear operator $\bar{U} : \mathbb{H} \rightarrow \mathbb{H}$ by $\bar{U}(H) = U M U^\dagger$ for every $M \in \mathbb{H}$. The operator \bar{U} is the adjoint representation of U .

The adjoint representation $U \mapsto \bar{U}$ defines a group homomorphism from $U(3, \mathbb{C})$ to $SO(8, \mathbb{R})$.

For U a Clifford+T operator, we will be interested in the matrix representation of \bar{U} in some convenient basis. Following [14], for a single-qutrit Pauli P , we set

$$P_\pm = \frac{P^\dagger \pm P}{\sqrt{\text{Tr}[(P^\dagger \pm P)^2]}}$$

in order to define a basis \mathcal{B} for \mathbb{H} .

► **Definition 32.** Let X and Z be the single-qutrit Pauli operators and let \mathbb{H} be the inner product space of 3×3 traceless Hermitian matrices. We define the orthogonal basis \mathcal{B} for \mathbb{H} as follows

$$\mathcal{B} = \{Z_+, X_+, (XZ)_+, (XZ^2)_+, Z_-, X_-, (XZ)_-, (XZ^2)_-\}.$$

If U is a Clifford+T operator, then the matrix for \bar{U} in the basis \mathcal{B} (ordered as in Definition 32) has several useful properties, as detailed in the following proposition, whose proof can be found in [14, Remark 4.15, Remark 4.18, and Proposition 4.20].

► **Proposition 33.** Let U be a 3×3 unitary matrix and assume that U can be exactly represented by an ancilla-free single-qutrit Clifford+T circuit. Then, in the basis \mathcal{B} , the operator \bar{U} has entries in the number ring \mathbb{A} . Write

$$\bar{U} = \left(\begin{array}{c|c} A & B \\ \hline C & D \end{array} \right)$$

where A , B , C , and D are 4×4 matrices. If the minimal T-count of U restricted to single-qutrit circuits is k , then the LDE of submatrix A is $2k$ and the following statements hold:

- If $k = 0$, then U is a Clifford operator.
- If $k > 0$, then up to generalized row and column permutations over \mathbb{Z}_3 ,

$$\rho_{2k}(A) \sim \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 2 & 2 & 2 \\ 0 & 2 & 2 & 2 \\ 0 & 2 & 2 & 2 \end{bmatrix} \quad \text{and} \quad \rho_{2k+1}(C) \sim \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 \end{bmatrix}.$$

We are now in a position to prove that R cannot be represented over the Clifford+ T gate set without using ancillas.

► **Proposition 34.** *The R gate cannot be represented by a single-qutrit ancilla-free Clifford+ T circuit.*

Proof. Direct computation yields

$$\overline{R} = \left(\begin{array}{c|c} A & B \\ \hline C & D \end{array} \right)$$

where

$$A = D = \frac{1}{3} \begin{pmatrix} 3 & 0 & 0 & 0 \\ 0 & -1 & 2 & 2 \\ 0 & 2 & -1 & 2 \\ 0 & 2 & 2 & -1 \end{pmatrix} \quad \text{and} \quad B = C = 0$$

Thus \overline{R} is a matrix over \mathbb{A} . The LDE of A is 6, and thus we compute

$$\rho_6(A) = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 2 & 2 & 2 \\ 0 & 2 & 2 & 2 \\ 0 & 2 & 2 & 2 \end{bmatrix} \quad \text{and} \quad \rho_7(C) = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}.$$

In particular, $\rho_7(C)$ is *not* equivalent up to generalized row/column permutations to the matrix

$$\begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 \end{bmatrix}.$$

Thus, R cannot be represented by a single-qutrit ancilla-free Clifford+ T circuit. ◀

