Complexity-Theoretic Foundations of Quantum Supremacy Experiments*

Scott Aaronson[1] and Lijie Chen[2]

1 **The University of Texas at Austin, Austin, TX, USA**
aaronson@cs.utexas.edu
2 **Tsinghua University, Beijing, China**
wjmzbmr@gmail.com

Abstract

In the near future, there will likely be special-purpose quantum computers with 40–50 high-quality qubits. This paper lays general theoretical foundations for how to use such devices to demonstrate "quantum supremacy": that is, a clear quantum speedup for *some* task, motivated by the goal of overturning the Extended Church-Turing Thesis as confidently as possible.

First, we study the hardness of sampling the output distribution of a random quantum circuit, along the lines of a recent proposal by the Quantum AI group at Google. We show that there's a natural average-case hardness assumption, which has nothing to do with sampling, yet implies that no polynomial-time classical algorithm can pass a statistical test that the quantum sampling procedure's outputs do pass. Compared to previous work – for example, on BosonSampling and IQP – the central advantage is that we can now talk directly about the observed outputs, rather than about the distribution being sampled.

Second, in an attempt to refute our hardness assumption, we give a new algorithm, inspired by Savitch's Theorem, for simulating a general quantum circuit with n qubits and depth d in polynomial space and $d^{O(n)}$ time. We then discuss why this and other known algorithms fail to refute our assumption.

Third, resolving an open problem of Aaronson and Arkhipov, we show that any strong quantum supremacy theorem – of the form "if approximate quantum sampling is classically easy, then the polynomial hierarchy collapses"– must be non-relativizing. This sharply contrasts with the situation for exact sampling.

Fourth, refuting a conjecture by Aaronson and Ambainis, we show that there is a sampling task, namely Fourier Sampling, with a *1 versus linear* separation between its quantum and classical query complexities.

Fifth, in search of a "happy medium"between black-box and non-black-box arguments, we study quantum supremacy relative to oracles in P/poly. Previous work implies that, if one-way functions exist, then quantum supremacy is possible relative to such oracles. We show, conversely, that *some* computational assumption is needed: if SampBPP = SampBQP and NP ⊆ BPP, then quantum supremacy is impossible relative to oracles with small circuits.

1998 ACM Subject Classification F.1.1 Models of Computation

Keywords and phrases computational complexity, quantum computing, quantum supremacy

Digital Object Identifier 10.4230/LIPIcs.CCC.2017.22

* Scott Aaronson is supported by a Vannevar Bush Faculty Fellowship from the US Department of Defense, and by the Simons Foundation "It from Qubit" Collaboration. Lijie Chen is supported in part by the National Basic Research Program of China Grant 2011CBA00300, 2011CBA00301, the National Natural Science Foundation of China Grant 61361136003.

32nd Computational Complexity Conference (CCC 2017).
Editor: Ryan O'Donnell; Article No. 22; pp. 22:1–22:67

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

1 Introduction

The *Extended Church-Turing Thesis*, or ECT, asserts that every physical process can be simulated by a deterministic or probabilistic Turing machine with at most polynomial overhead. Since the 1980s – and certainly since the discovery of Shor's algorithm [55] in the 1990s – computer scientists have understood that quantum mechanics might refute the ECT in principle. Today, there are actual experiments being planned (e.g., [19]) with the goal of severely challenging the ECT in practice. These experiments don't yet aim to build full, fault-tolerant, universal quantum computers, but "merely" to demonstrate *some* quantum speedup over the best known or conjectured classical algorithms, for some possibly-contrived task, as confidently as possible. In other words, the goal is to answer the skeptics [41, 38] who claim that genuine quantum speedups are either impossible in theory, or at any rate, are hopelessly out of reach technologically. Recently, the term "quantum supremacy" has come into vogue for such experiments, footnoteAs far as we know, the first person to use the term in print was John Preskill [47]. although the basic goal goes back several decades, to the beginning of quantum computing itself.

Before going further, we should address some common misunderstandings about quantum supremacy.

The ECT is an asymptotic claim, which of course means that *no* finite experiment could render a decisive verdict on it, even in principle. But this hardly makes experiments irrelevant. If

1. a quantum device performed some task (say) 10^{15} times faster than a highly optimized simulation written by "adversaries" and running on a classical computing cluster, with the quantum/classical gap appearing to increase exponentially with the instance size across the whole range tested, and
2. this observed performance closely matched theoretical results that *predicted* such an exponential quantum speedup for the task in question, and
3. all other consistency checks passed (for example: removing quantum behavior from the experimental device destroyed the observed speedup),

this would obviously "raise the stakes" for anyone who still believed the ECT! Indeed, when some quantum computing researchers have criticized previous claims to have experimentally achieved quantum speedups (see, e.g., [2]), it has typically been on the ground that, in those researchers' view, the experiments failed to meet one or more of the conditions above.

It's sometimes claimed that any molecule in Nature or the laboratory, for which chemists find it computationally prohibitive to solve the Schrödinger equation and calculate its ground state, *already* provides an example of "quantum supremacy."The idea, in other words, is that such a molecule constitutes a "useful quantum computer, for the task of simulating itself."

For us, the central problem with this idea is that in theoretical computer science, we care less about individual *instances* than about solving *problems* (i.e., infinite collections of instances) in a more-or-less uniform way. For any one molecule, the difficulty in simulating it classically *might* reflect genuine asymptotic hardness, but it might also reflect other issues (e.g., a failure to exploit special structure in the molecule, or the same issues of modeling error, constant-factor overheads, and so forth that arise even in simulations of classical physics).

Thus, while it's possible that complex molecules could form the basis for a convincing quantum supremacy demonstration, we believe more work would need to be done. In particular, one would want a device that could synthesize *any* molecule in some theoretically infinite class – and one would then want complexity-theoretic evidence that the general

problem, of simulating a given molecule from that class, is asymptotically hard for a classical computer. And in such a case, it would seem more natural to call the synthesis machine the "quantum computer, textquotedblright rather than the molecules themselves!

In summary, we regard quantum supremacy as a central milestone for quantum computing that hasn't been reached yet, but that might be reached in the near future. This milestone is essentially negative in character: it has no obvious signature of the sort familiar to experimental physics, since it simply amounts to the *nonexistence* of an efficient classical algorithm to simulate a given quantum process. For that reason, the tools of theoretical computer science will be essential to understand when quantum supremacy has or hasn't been achieved. So in our view, even if it were uninteresting as TCS, there would still be an urgent need for TCS to contribute to the discussion about which quantum supremacy experiments to do, how to verify their results, and what should count as convincing evidence that classical simulation is hard. Happily, it turns out that there *is* a great deal here of intrinsic TCS interest as well.

1.1 Supremacy from Sampling

In recent years, a realization has crystallized that, if our goal is to demonstrate quantum supremacy (rather than doing anything directly useful), then there are good reasons to shift our attention from decision and function problems to *sampling* problems: that is, problems where the goal is to sample an n-bit string, either exactly or approximately, from a desired probability distribution.

A first reason for this is that demonstrating quantum supremacy via a sampling problem doesn't appear to require the full strength of a universal quantum computer. Indeed, there are now at least a half-dozen proposals [57, 23, 3, 44, 37, 29, 11] for special-purpose devices that could efficiently solve sampling problems believed to be classically intractable, *without* being able to solve every problem in the class BQP, or for that matter even every problem in P. Besides their intrinsic physical and mathematical interest, these intermediate models might be easier to realize than a universal quantum computer. In particular, because of their simplicity, they might let us avoid the need for the full machinery of quantum fault-tolerance [12]: something that adds a theoretically polylogarithmic but enormous-in-practice overhead to quantum computation. Thus, many researchers now expect that the first convincing demonstration of quantum supremacy will come via this route.

A second reason to focus on sampling problems is more theoretical: in the present state of complexity theory, we can arguably be *more* confident that certain quantum sampling problems really are classically hard, than we are that *factoring* (for example) is classically hard, or even that $\mathsf{BPP} \neq \mathsf{BQP}$. Already in 2002, Terhal and DiVincenzo [57] noticed that, while constant-depth quantum circuits can't solve any classically intractable decision problems, footnoteThis is because any qubit output by such a circuit depends on at most a constant number of input qubits. they nevertheless have a curious power: namely, they can sample probability distributions that can't be sampled in classical polynomial time, unless $\mathsf{BQP} \subseteq \mathsf{AM}$, which would be a surprising inclusion of complexity classes. Then, in 2004, Aaronson showed that $\mathsf{PostBQP} = \mathsf{PP}$, where $\mathsf{PostBQP}$ means BQP with the ability to postselect on exponentially-unlikely measurement outcomes. This had the immediate corollary that, if there's an efficient classical algorithm to sample the output distribution of an arbitrary quantum circuit – or for that matter, any distribution whose probabilities are multiplicatively close to the correct ones – then

$$\mathsf{PP} = \mathsf{PostBQP} = \mathsf{PostBPP} \subseteq \mathsf{BPP}^{\mathsf{NP}}.$$

By Toda's Theorem [58], this implies that the polynomial hierarchy collapses to the third level.

Related to that, in 2009, Aaronson [1] showed that, while it was (and remains) a notorious open problem to construct an oracle relative to which $\mathsf{BQP} \not\subset \mathsf{PH}$, one can construct oracular *sampling* and *relation* problems that are solvable in quantum polynomial time, but that are provably not solvable in randomized polynomial time augmented with a PH oracle.

Then, partly inspired by that oracle separation, Aaronson and Arkhipov [7, 3] proposed BosonSampling: a model that uses identical photons traveling through a network of beamsplitters and phaseshifters to solve classically hard sampling problems. Aaronson and Arkhipov proved that a polynomial-time exact classical simulation of BosonSampling would collapse PH. They also gave a plausible conjecture implying that even an *approximate* simulation would have the same consequence. Around the same time, Bremner, Jozsa, and Shepherd [23] independently proposed the Commuting Hamiltonians or IQP ("Instantaneous Quantum Polynomial-Time") model, and showed that it had the same property, that exact classical simulation would collapse PH. Later, Bremner, Montanaro, and Shepherd [24, 25] showed that, just like for BosonSampling, there are plausible conjectures under which even a fast classical *approximate* simulation of the IQP model would collapse PH.

Since then, other models have been proposed with similar behavior. To take a few examples: Farhi and Harrow [29] showed that the so-called Quantum Approximate Optimization Algorithm, or QAOA, can sample distributions that are classically intractable unless PH collapses. Morimae, Fujii, and Fitzsimons [44] showed the same for the so-called One Clean Qubit or DQC1 model, while Jozsa and Van den Nest [37] showed it for stabilizer circuits with magic initial states and nonadaptive measurements, and Aaronson et al. [11] showed it for a model based on integrable particle scattering in $1+1$ dimensions. In retrospect, the constant-depth quantum circuits considered by Terhal and DiVincenzo [57] also have the property that fast exact classical simulation would collapse PH.

Within the last four years, quantum supremacy via sampling has made the leap from complexity theory to a serious experimental prospect. For example, there have by now been many small-scale demonstrations of BosonSampling in linear-optical systems, with the current record being a 6-photon experiment by Carolan et al. [27]. To scale up to (say) 30 or 40 photons – as would be needed to make a classical simulation of the experiment suitably difficult – seems to require more reliable single-photon sources than exist today. But some experts (e.g., [50, 46]) are optimistic that optical multiplexing, superconducting resonators, or other technologies currently under development will lead to such photon sources. In the meantime, as we mentioned earlier, Boixo et al. [19] have publicly detailed a plan, currently underway at Google, to perform a quantum supremacy experiment involving random circuits applied to a 2D array of 40-50 coupled superconducting qubits. So far, the group at Google has demonstrated the preparation and measurement of entangled states on a linear array of 9 superconducting qubits [39].

1.2 Theoretical Challenges

Despite the exciting recent progress in both theory and experiment, some huge conceptual problems have remained about sampling-based quantum supremacy. These problems are not specific to any one quantum supremacy proposal (such as BosonSampling, IQP, or random quantum circuits), but apply with minor variations to all of them.

Verification of Quantum Supremacy Experiments. From the beginning, there was the problem of *how to verify the results* of a sampling-based quantum supremacy experiment. In contrast to (say) factoring and discrete log, for sampling tasks such as BosonSampling, it seems unlikely that there's any NP witness certifying the quantum experiment's output, let alone an NP witness that's also the experimental output itself. Rather, for the sampling tasks, not only simulation but even verification might need classical exponential time. Yet, while no one has yet discovered a general way around this, footnoteIn principle, one could use so-called *authenticated quantum computing* [13, 26], but the known schemes for that might be much harder to realize technologically than a basic quantum supremacy experiment, and in any case, they all presuppose the validity of quantum mechanics. it's far from the fatal problem that some have imagined. The reason is simply that experiments can and will target a "sweet spot, textquotedblright of (say) 40–50 qubits, for which classical simulation and verification of the results is *difficult but not impossible*.

Still, the existing verification methods have a second drawback. Namely, once we've fixed a specific verification test for sampling from a probability distribution $\mathcal{D}$, we ought to consider, not merely all classical algorithms that sample exactly or approximately from $\mathcal{D}$, but *all classical algorithms that output anything that passes the verification test.* To put it differently, we ought to talk not about the sampling problem itself, but about an associated *relation problem*: that is, a problem where the goal is to produce any output that satisfies a given condition.

As it happens, in 2011, Aaronson [8] proved an extremely general connection between sampling problems and relation problems. Namely, given any approximate sampling problem S, he showed how to define a relation problem R_S such that, for every "reasonable" model of computation (classical, quantum, etc.), R_S is efficiently solvable in that model if and only if S is. This had the corollary that

$$\mathsf{SampBPP} = \mathsf{SampBQP} \iff \mathsf{FBPP} = \mathsf{FBQP},$$

where SampBPP and SampBQP are the classes of approximate sampling problems solvable in polynomial time by randomized and quantum algorithms respectively, and FBPP and FBQP are the corresponding classes of relation problems. Unfortunately, Aaronson's construction of R_S involved Kolmogorov complexity: basically, one asks for an m-tuple of strings, $\langle x_1, \ldots, x_m \rangle$, such that

$$K(x_1, \ldots, x_m) \geq \log_2 \frac{1}{p_1 \cdots p_m} - O(1),$$

where p_i is the desired probability of outputting x_i in the sampling problem. And of course, verifying such a condition is extraordinarily difficult, even more so than calculating the probabilities $p_1, \ldots, p_m$.[1] For this reason, it's strongly preferable to have a condition that talks only about the largeness of the p_i's, and not about the algorithmic randomness of the x_i's. But then hardness for the sampling problem no longer necessarily implies hardness for the relation problem, so a new argument is needed.

Supremacy Theorems for Approximate Sampling. A second difficulty is that any quantum sampling device is subject to noise and decoherence. Ultimately, of course, we'd like hardness

[1] Furthermore, this is true even if we substitute a resource-bounded Kolmogorov complexity, as Aaronson's result allows.

results for quantum sampling that apply even in the presence of experimentally realistic errors. Very recently, Bremner, Montanaro, and Shepherd [25] and Fujii [31] have taken some promising initial steps in that direction. But even if we care only about the *smallest* "experimentally reasonable" error – namely, an error that corrupts the output distribution $\mathcal{D}$ to some other distribution $\mathcal{D}'$ that's ε-close to $\mathcal{D}$ in variation distance – Aaronson and Arkhipov [3] found that we already engage substantial new open problems in complexity theory, if we want evidence for classical hardness. So for example, their hardness argument for approximate BosonSampling depended on the conjecture that there's no $\mathsf{BPP}^{\mathsf{NP}}$ algorithm to estimate the permanent of an i.i.d. Gaussian matrix $A \sim N(0,1)_{\mathbb{C}}^{n\times n}$, with high probability over the choice of A.

Of course, one could try to close that loophole by proving that this Gaussian permanent estimation problem is #P-hard, which is indeed a major challenge that Aaronson and Arkhipov left open. But this situation also raises more general questions. For example, is there an implication of the form "if SampBPP = SampBQP, then PH collapses, textquotedblright where again SampBPP and SampBQP are the *approximate* sampling versions of BPP and BQP respectively? Are there oracles relative to which such an implication does *not* hold?

Quantum Supremacy Relative to Oracles. A third problem goes to perhaps the core issue of complexity theory (both quantum and classical): namely, we don't at present have a proof of P $\neq$ PSPACE, much less of BPP $\neq$ BQP or SampBPP $\neq$ SampBQP, less still of the hardness of specific problems like factoring or estimating Gaussian permanents. So what reason do we have to believe that *any* of these problems are hard? Part of the evidence has always come from oracle results, which we often *can* prove unconditionally. Particularly in quantum complexity theory, oracle separations can already be highly nontrivial, and give us a deep intuition for why all the "standard"algorithmic approaches fail for some problem.

On the other hand, we also know, from results like IP = PSPACE [54], that oracle separations can badly mislead us about what happens in the unrelativized world. Generally speaking, we might say, relying on an oracle separation is more dangerous, the less the oracle function resembles what would actually be available in an explicit problem.[2]

In the case of sampling-based quantum supremacy, we've known strong oracle separations since early in the subject. Indeed, in 2009, Aaronson [1] showed that Fourier Sampling – a quantumly easy sampling problem that involves only a *random* oracle – requires classical exponential time, and for that matter, sits outside the entire polynomial hierarchy. But of course, in real life random oracles are unavailable. So a question arises: can we say anything about the classical hardness of Fourier Sampling with a *pseudorandom* oracle? More broadly, what hardness results can we prove for quantum sampling, relative to oracles that are efficiently computable? Here, we imagine that an algorithm doesn't have access to a succinct representation of the oracle function f, but it does know that a succinct representation *exists* (i.e., that $f \in \mathsf{P/poly}$). Under that assumption, is there any hope of proving an *unconditional* separation between quantum and classical sampling? If not, then can we at least prove quantum supremacy under weaker (or more "generic") assumptions than would be needed in the purely computational setting?

[2] Indeed, the *algebrization barrier* of Aaronson and Wigderson [6] was based on precisely this insight: namely, if we force oracles to be "more realistic, textquotedblright by demanding (in that case) that they come equipped with algebraic extensions of whichever Boolean functions they represent, then many previously non-relativizing results become relativizing.

1.3 Our Contributions

In this paper, we address all three of the above challenges. Our results might look wide-ranging, but they're held together by a single thread: namely, *the quest to understand the classical hardness of quantum approximate sampling problems, and especially the meta-question of under which computational assumptions such hardness can be proven.* We'll be interested in both "positive" results, of the form "quantum sampling problem X is classically hard under assumption Y, textquotedblright and "negative" results, of the form "proving the classical hardness of X *requires* assumption Y."Also, we'll be less concerned with specific proposals such as BosonSampling, than simply with the general task of approximately sampling the output distribution of a given quantum circuit C. Fortuitously, though, our focus on quantum circuit sampling will make some of our results an excellent fit to currently planned experiments – most notably, those at Google [19], which will involve random quantum circuits on a 2D square lattice of 40 to 50 superconducting qubits. Even though we won't address the details of those or other experiments, our results (together with other recent work [19, 25]) can help to inform the experiments – for example, by showing how the circuit depth, the verification test applied to the outputs, and other design choices affect the strength of the computational assumptions that are necessary and sufficient to conclude that quantum supremacy has been achieved.

We have five main results.

The Hardness of Quantum Circuit Sampling. Our first result, in Section 3, is about the hardness of sampling the output distribution of a random quantum circuit, along the general lines of the planned Google experiment. Specifically, we propose a simple verification test to apply to the outputs of a random quantum circuit. We then analyze the classical hardness of generating *any* outputs that pass that test.

More concretely, we study the following basic problem:

▶ **Problem 1** (HOG, or Heavy Output Generation). *Given as input a random quantum circuit C (drawn from some suitable ensemble), generate output strings $x_1, \ldots, x_k$, at least a $2/3$ fraction of which have greater than the median probability in C's output distribution.*

HOG is a relation problem, for which we can verify a claimed solution in classical exponential time, by calculating the ideal probabilities $p_{x_1}, \ldots, p_{x_k}$ for each x_i to be generated by C, and then checking whether enough of the p_{x_i}'s are greater than the median value (which we can estimate analytically to extremely high confidence). Furthermore, HOG is easy to solve on a quantum computer, with overwhelming success probability, by the obvious strategy of just running C over and over and collecting k of its outputs.[3]

It certainly seems plausible that HOG is exponentially hard for a classical computer. But we ask: under what assumption could that hardness be proven? To address that question, we propose a new hardness assumption:

▶ **Assumption 1** (QUATH, or the QUAntum THreshold assumption). *There is no polynomial-time classical algorithm that takes as input a description of a random quantum circuit C, and that guesses whether $|\langle 0^n|C|0^n\rangle|^2$ is greater or less than the median of all 2^n of the $|\langle 0^n|C|x\rangle|^2$ values, with success probability at least $\frac{1}{2} + \Omega\left(\frac{1}{2^n}\right)$ over the choice of C.*

[3] Heuristically, one expects the p_{x_i}'s to be exponentially distributed random variables, which one can calculate implies that a roughly $\frac{1+\ln 2}{2} \approx 0.847$ fraction of the outputs will have probabilities exceeding the median value.

Our first result says that if QUATH is true, then HOG is hard. While this might seem nearly tautological, the important point here is that QUATH makes no reference to sampling or relation problems. Thus, we can now shift our focus from sampling algorithms to algorithms that simply estimate amplitudes, with a minuscule advantage over random guessing.

New Algorithms to Simulate Quantum Circuits. But given what a tiny advantage $\Omega\left(2^{-n}\right)$ is, why would anyone even conjecture that QUATH might be true? This brings us to our second result, in Section 4, which is motivated by the attempt to refute QUATH. We ask: what *are* the best classical algorithms to simulate an arbitrary quantum circuit? For special quantum circuits (e.g., those with mostly Clifford gates and a few T gates [22]), there's been exciting recent progress on improved exponential-time simulation algorithms, but for arbitrary quantum circuits, one might think there isn't much to say. Nevertheless, we *do* find something basic to say that, to our knowledge, had been overlooked earlier.

For a quantum circuit with n qubits and m gates, there are two obvious simulation algorithms. The first, which we could call the "Schrödinger" algorithm, stores the entire state vector in memory, using $\sim m2^n$ time and $\sim 2^n$ space. The second, which we could call the "Feynman" algorithm, calculates an amplitude as a sum of terms, using $\sim 4^m$ time and $\sim m+n$ space, as in the proof of $\mathsf{BQP} \subseteq \mathsf{P}^{\#\mathsf{P}}$ [18].

Now typically $m \gg n$, and the difference between m and n could matter enormously in practice. For example, in the planned Google setup, n will be roughly 40 or 50, while m will ideally be in the thousands. Thus, 2^n time is reasonable whereas 4^m time is not. So a question arises:

- *When $m \gg n$, is there a classical algorithm to simulate an n-qubit, m-gate quantum circuit using both $\mathrm{poly}(m,n)$ space and much less than $\exp(m)$ time – ideally, more like $\exp(n)$?*

We show an affirmative answer. In particular, inspired by the proof of Savitch's Theorem [52], we give a recursive, sum-of-products algorithm that uses $\mathrm{poly}(m,n)$ space and $m^{O(n)}$ time – or better yet, $d^{O(n)}$ time, where d is the circuit depth. We also show how to improve the running time further for quantum circuits subject to nearest-neighbor constraints, such as the superconducting systems currently under development. Finally, we show the existence of a "smooth tradeoff" between our algorithm and the 2^n-memory Schrödinger algorithm. Namely, starting with the Schrödinger algorithm, for every desired halving of the memory usage, one can multiply the running time by an additional factor of $\sim d$.

We hope our algorithm finds some applications in quantum simulation. In the meantime, though, the key point for this paper is that neither the Feynman algorithm, nor the Schrödinger algorithm, nor our new recursive algorithm come close to refuting QUATH. The Feynman algorithm fails to refute QUATH because it yields only a $1/\exp(m)$ advantage over random guessing, rather than a $1/2^n$ advantage.[4] The Schrödinger and recursive algorithms have much closer to the "correct" 2^n running time, but they also fail to refute QUATH because they don't calculate amplitudes as straightforward sums, so don't lead to polynomial-time guessing algorithms at all. Thus, in asking whether we can falsify QUATH, in some sense we're asking how far we can go in combining the advantages of all these algorithms. This might, in turn, connect to longstanding open problems about the optimality of Savitch's Theorem itself (e.g., L versus NL).

[4] Note that the Feynman algorithm can also be interpreted as a PP algorithm.

Interestingly, our analysis of quantum circuit simulation algorithms explains why this paper's hardness argument for quantum circuit sampling, based on QUATH, would *not* have worked for quantum supremacy proposals such as BosonSampling or IQP. It works only for the more general problem of quantum circuit sampling. The reason is that for the latter, unlike for BosonSampling or IQP, there exists a parameter $m \gg n$ (namely, the number of gates) that controls the advantage that a polynomial-time classical algorithm can achieve over random guessing, even while n controls the number of possible outputs. Our analysis also underscores the importance of taking $m \gg n$ in experiments meant to show quantum supremacy, and it provides some guidance to experimenters about the crucial question of what *circuit depth* they need for a convincing quantum supremacy demonstration.

Note that, the greater the required depth, the more protected against decoherence the qubits need to be. But the tradeoff is that the depth must be high enough that simulation algorithms that exploit limited entanglement, such as those based on tensor networks, are ruled out. Beyond that requirement, our $d^{O(n)}$ simulation algorithm gives some information about how much additional hardness one can purchase for a given increase in depth.

Strong Quantum Supremacy Theorems Must Be Non-Relativizing. Next, in Section 5, we switch our attention to a meta-question. Namely, what sorts of complexity-theoretic evidence we could possibly hope to offer for $\mathsf{SampBPP} \neq \mathsf{SampBQP}$: in other words, for quantum computers being able to solve approximate sampling problems that are hard classically? By Aaronson's sampling/searching equivalence theorem [8], any such evidence would *also* be evidence for $\mathsf{FBPP} \neq \mathsf{FBQP}$ (where FBPP and FBQP are the corresponding classes of relation problems), and vice versa.

Of course, an unconditional proof of these separations is out of the question right now, since it would imply $\mathsf{P} \neq \mathsf{PSPACE}$. Perhaps the next best thing would be to show that, if $\mathsf{SampBPP} = \mathsf{SampBQP}$, then the polynomial hierarchy collapses. This latter is *not* out of the question: as we said earlier, we already know, by a simple relativizing argument, that an equivalence between quantum and classical *exact* sampling implies the collapse $\mathsf{P}^{\#\mathsf{P}} = \mathsf{PH} = \mathsf{BPP}^{\mathsf{NP}}$. Furthermore, in their work on BosonSampling, Aaronson and Arkhipov [3] formulated a #P-hardness conjecture – namely, their so-called *Permanent of Gaussians Conjecture*, or PGC – that if true, would imply a generalization of that collapse to the physically relevant case of approximate sampling. More explicitly, Aaronson and Arkhipov showed that if the PGC holds, then

$$\mathsf{SampBPP} = \mathsf{SampBQP} \Longrightarrow \mathsf{P}^{\#\mathsf{P}} = \mathsf{BPP}^{\mathsf{NP}}. \tag{1}$$

They went on to propose a program for proving the PGC, by exploiting the random self-reducibility of the permanent. On the other hand, Aaronson and Arkhipov also explained in detail why new ideas would be needed to complete that program, and the challenge remains open.

Subsequently, Bremner, Montanaro, and Shepherd [24, 25] gave analogous #P-hardness conjectures that, if true, would *also* imply the implication (1), by going through the IQP model rather than through BosonSampling.

Meanwhile, nearly two decades ago, Fortnow and Rogers [30] exhibited an oracle relative to which $\mathsf{P} = \mathsf{BQP}$ and yet the polynomial hierarchy is infinite. In other words, they showed that any proof of the implication

$$\mathsf{P} = \mathsf{BQP} \Longrightarrow \mathsf{PH} \text{ collapses}$$

would have to be non-relativizing. Unfortunately, their construction was extremely specific to languages (i.e., total Boolean functions), and didn't even rule out the possibility that the

implication

$$\mathsf{PromiseBPP} = \mathsf{PromiseBQP} \implies \mathsf{PH collapses}$$

could be proven in a relativizing way. Thus, Aaronson and Arkhipov [3, see Section 10] raised the question of which quantum supremacy theorems hold relative to all oracles.

In Section 5, we fill in the final piece needed to resolve their question, by constructing an oracle A relative to which $\mathsf{SampBPP}^A = \mathsf{SampBQP}^A$ and yet PH^A is infinite. In other words, we show that *any strong supremacy theorem for quantum sampling, along the lines of what Aaronson and Arkhipov [3] and Bremner, Montanaro, and Shepherd [24, 25] were seeking, must use non-relativizing techniques.* In that respect, the situation with approximate sampling is extremely different from that with exact sampling.

Perhaps it's no surprise that one would need non-relativizing techniques to prove a strong quantum supremacy theorem. In fact, Aaronson and Arkhipov [3] were originally led to study BosonSampling precisely because of the connection between bosons and the permanent function, and the hope that one could therefore exploit the famous non-relativizing properties of the permanent to prove hardness. All the same, this is the first time we have explicit confirmation that non-relativizing techniques will be needed.

Maximal Quantum Supremacy for Black-Box Sampling and Relation Problems. In Section 6, we turn our attention to the black-box model, and specifically to the question: *what are the largest possible separations between randomized and quantum query complexities for any approximate sampling or relation problem?* Here we settle another open question. In 2015, Aaronson and Ambainis [9] studied Fourier Sampling, in which we're given access to a Boolean function $f : \{0,1\}^n \rightarrow \{0,1\}$, and the goal is to sample a string z with probability $\widehat{f}(z)^2$, where $\widehat{f}$ is the Boolean Fourier transform of f, normalized so that $\sum_z \widehat{f}(z)^2 = 1$. This problem is trivially solvable by a quantum algorithm with only 1 query to f. By contrast, Aaronson and Ambainis showed that there exists a constant $\varepsilon > 0$ such that any classical algorithm that solves Fourier Sampling, to accuracy ε in variation distance, requires $\Omega(2^n/n)$ queries to f. They conjectured that this lower bound was tight.

Here we refute that conjecture, by proving a $\Omega(2^n)$ lower bound on the randomized query complexity of Fourier Sampling, as long as ε is sufficiently small (say, $\frac{1}{40000}$). This implies that, for approximate sampling problems, the gap between quantum and randomized query complexities can be as large as imaginable: namely, *1 versus linear (!)*.[5] This sharply contrasts with the case of partial Boolean functions, for which Aaronson and Ambainis [9] showed that any N-bit problem solvable with k quantum queries is also solvable with $O\left(N^{1-1/2k}\right)$ randomized queries, and hence a constant versus linear separation is impossible. Thus, our result helps once again to underscore the advantage of sampling problems over decision problems for quantum supremacy experiments. Given the extremely close connection between Fourier Sampling and the IQP model [23], our result also provides some evidence that classically simulating an n-qubit IQP circuit, to within constant error in variation distance, is about as hard as can be: it might literally require $\Omega(2^n)$ time.

Aaronson and Ambainis [9] didn't directly address the natural relational version of Fourier Sampling, which Aaronson [1] had called Fourier Fishing in 2009. In Fourier Fishing,

[5] We have learned (personal communication) that recently, and independently of us, Ashley Montanaro has obtained a communication complexity result that implies this result as a corollary.

the goal is to output any string z such that $\widehat{f}(z)^2 \geq 1$, with nontrivial success probability. Unfortunately, the best lower bound on the randomized query complexity of Fourier Fishing that follows from [1] has the form $2^{n^{\Omega(1)}}$. As a further contribution, in Section 6 we give a lower bound of $\Omega(2^n/n)$ on the randomized query complexity of Fourier Fishing, which both simplifies and subsumes the $\Omega(2^n/n)$ lower bound for Fourier Sampling by Aaronson and Ambainis [9] (which, of course, we also improve to $\Omega(2^n)$ in this paper).

Quantum Supremacy Relative to Efficiently-Computable Oracles. In Section 7, we ask a new question: when proving quantum supremacy theorems, can we "interpolate" between the black-box setting of Sections 5 and 6, and the non-black-box setting of Sections 3 and 4? In particular, what happens if we consider quantum sampling algorithms that can access an oracle, *but* we impose a constraint that the oracle has to be "physically realistic"? One natural requirement here is that the oracle function f be computable in the class $\mathsf{P/poly}$:[6] in other words, that there are polynomial-size circuits for f, which we imagine that our sampling algorithms (both quantum and classical) can call as subroutines. If the sampling algorithms *also* had access to explicit descriptions of the circuits, then we'd be back in the computational setting, where we already know that there's no hope at present of proving quantum supremacy unconditionally. But what if our sampling algorithms know only that small circuits for f exist, without knowing what they are? Could quantum supremacy be proven unconditionally *then*?

We give a satisfying answer to this question. First, by adapting constructions due to Zhandry [61] and (independently) Servedio and Gortler [53], we show that if one-way functions exist, then there are oracles $A \in \mathsf{P/poly}$ such that $\mathsf{BPP}^A \neq \mathsf{BQP}^A$, and indeed even $\mathsf{BQP}^A \not\subset \mathsf{SZK}^A$. (Here and later, the one-way functions only need to be hard to invert classically, not quantumly.)

Note that, in the unrelativized world, there seems to be no hope at present of proving $\mathsf{BPP} \neq \mathsf{BQP}$ under any hypothesis nearly as weak as the existence of one-way functions. Instead one has to assume the one-wayness of extremely *specific* functions, for example those based on factoring or discrete log.

Second, and more relevant to near-term experiments, we show that if there exist one-way functions that take at least subexponential time to invert, then there are Boolean functions $f \in \mathsf{P/poly}$ such that approximate Fourier Sampling on those f's requires classical exponential time. In other words: within our "physically realistic oracle" model, there are feasible-looking quantum supremacy experiments, along the lines of the IQP proposal [23], such that a very standard and minimal cryptographic assumption is enough to prove the hardness of simulating those experiments classically.

Third, we show that the above two results are essentially optimal, by proving a converse result: that even in our $\mathsf{P/poly}$ oracle model, *some* computational assumption is still needed to prove quantum supremacy. The precise statement is this: if $\mathsf{SampBPP} = \mathsf{SampBQP}$ and $\mathsf{NP} \subseteq \mathsf{BPP}$, then $\mathsf{SampBPP}^A = \mathsf{SampBQP}^A$ for all $A \in \mathsf{P/poly}$. Or equivalently: if we want to separate quantum from classical approximate sampling relative to efficiently computable oracles, then we need to assume *something* about the unrelativized world: either $\mathsf{SampBPP} \neq \mathsf{SampBQP}$ (in which case we wouldn't even need an oracle), or else $\mathsf{NP} \not\subset \mathsf{BPP}$ (which is closely related to the assumption we *do* make, namely that one-way functions exist).

So to summarize, we've uncovered a "smooth tradeoff" between the model of computation and the hypothesis needed for quantum supremacy. Relative to *some* oracle (and even a

[6] More broadly, we could let f be computable in $\mathsf{BQP/poly}$, but this doesn't change the story too much.

random oracle), we can prove $\mathsf{SampBPP} \neq \mathsf{SampBQP}$ unconditionally. Relative to some efficiently computable oracle, we can prove $\mathsf{SampBPP} \neq \mathsf{SampBQP}$, but only under a weak computational assumption, like the existence of one-way functions. Finally, with no oracle, we can currently prove $\mathsf{SampBPP} \neq \mathsf{SampBQP}$ only under special assumptions, such as factoring being hard, or the permanents of Gaussian matrices being hard to approximate in $\mathsf{BPP}^{\mathsf{NP}}$, or our QUATH assumption. Perhaps eventually, we'll be able to prove $\mathsf{SampBPP} \neq \mathsf{SampBQP}$ under the sole assumption that PH is infinite, which would be a huge step forward – but at any rate we'll need *some* separation of classical complexity classes.[7]

One last remark: the idea of comparing complexity classes relative to $\mathsf{P/poly}$ oracles seems quite natural even apart from its applications to quantum supremacy. So in Appendix A, we take an initial stab at exploring the implications of that idea for other central questions in complexity theory. In particular, we prove the surprising result there that $\mathsf{P}^A = \mathsf{BPP}^A$ for all oracles $A \in \mathsf{P/poly}$, *if and only if* the derandomization hypothesis of Impagliazzo and Wigderson [36] holds (i.e., there exists a function in E with $2^{\Omega(n)}$ circuit complexity). In our view, this helps to clarify Impagliazzo and Wigderson's theorem itself, by showing precisely in what way their circuit lower bound hypothesis is stronger than the desired conclusion $\mathsf{P} = \mathsf{BPP}$. We also show that, if there are quantumly-secure one-way functions, then there exists an oracle $A \in \mathsf{P/poly}$ such that $\mathsf{SZK}^A \not\subset \mathsf{BQP}^A$.

1.4 Techniques

In our view, the central contributions of this work lie in the creation of new questions, models, and hardness assumptions (such as QUATH and quantum supremacy relative to $\mathsf{P/poly}$ oracles), as well as in basic observations that somehow weren't made before (such as the sum-products algorithm for simulating quantum circuits) – all of it motivated by the goal of using complexity theory to inform ongoing efforts in experimental physics to test the Extended Church-Turing Thesis. While some of our proofs are quite involved, by and large the proof techniques are ones that will be familiar to complexity theorists. Even so, it seems appropriate to say a few words about techniques here.

To prove, in Section 3, that "if QUATH is true, then HOG is hard, textquotedblright we give a fairly straightforward reduction: namely, we assume the existence of a polynomial-time classical algorithm to find high-probability outputs of a given quantum circuit C. We then use that algorithm (together with a random self-reduction trick) to guess the magnitude of a particular transition amplitude, such as $\langle 0^n|C|0^n\rangle$, with probability slightly better than chance, which is enough to refute QUATH.

One technical step is to show that, with $\Omega(1)$ probability, the distribution over n-bit strings sampled by a random quantum circuit C is far from the uniform distribution. But not only can this be done, we show that it can be done by examining only the very last gate of C, and ignoring all other gates! A challenge that we leave open is to improve this, to show that the distribution sampled by C is far from uniform, not merely with $\Omega(1)$ probability, but with $1 - 1/\exp(n)$ probability. In Appendix E, we present numerical evidence for this conjecture, and indeed for a stronger conjecture, that the probabilities appearing in the output distribution of a random quantum circuit behave like independent, exponentially-distributed random variables. (We note that Brandao, Harrow and Horodecki [21] recently proved a closely-related result, which unfortunately is not quite strong enough for our purposes.)

In Section 4, to give our polynomial-space, $d^{O(n)}$-time classical algorithm for simulating an n-qubit, depth-d quantum circuit C, we use a simple recursive strategy, reminiscent of

[7] Unless, of course, someone were to separate P from PSPACE unconditionally!

Savitch's Theorem. Namely, we slice the circuit into two layers, C_1 and C_2, of depth $d/2$ each, and then express a transition amplitude $\langle x|C|z\rangle$ of interest to us as

$$\langle x|C|z\rangle = \sum_{y\in\{0,1\}^n} \langle x|C_1|y\rangle\langle y|C_2|z\rangle.$$

We then compute each $\langle x|C_1|y\rangle$ and $\langle y|C_2|z\rangle$ by recursively slicing C_1 and C_2 into layers of depth $d/4$ each, and so on. What takes more work is to obtain a further improvement if C has only nearest-neighbor interactions on a grid graph – for that, we use a more sophisticated divide-and-conquer approach – and also to interpolate our recursive algorithm with the 2^n-space Schrödinger simulation, in order to make the best possible use of whatever memory is available.

Our construction, in Section 5, of an oracle relative to which SampBPP = SampBQP and yet PH is infinite involves significant technical difficulty. As a first step, we can use a PSPACE oracle to collapse SampBPP with SampBQP, and then use one of many known oracles (or, by the recent breakthrough of Rossman, Servedio, and Tan [49], even a *random* oracle) to make PH infinite. The problem is that, if we do this in any naïve way, then the oracle that makes PH infinite will also re-separate SampBPP and SampBQP, for example because of the approximate Fourier Sampling problem. Thus, we need to *hide* the oracle that makes PH infinite, in such a way that a PH algorithm can still find the oracle (and hence, PH is still infinite), but a SampBQP algorithm can't find it with any non-negligible probability – crucially, not even if the SampBQP algorithm's input x provides a clue about the oracle's location. Once one realizes that these are the challenges, one then has about seven pages of work to ensure that SampBPP and SampBQP remain equal, relative to the oracle that one has constructed. Incidentally, we know that this equivalence can't possibly hold for *exact* sampling, so *something* must force small errors to arise when the SampBPP algorithm simulates the SampBQP one. That something is basically the tiny probability that the quantum algorithm will succeed at finding the hidden oracle, which however can be upper-bounded using quantum-mechanical linearity.

In Section 6, to prove a $\Omega(2^n)$ lower bound on the classical query complexity of approximate Fourier Sampling, we use the same basic strategy that Aaronson and Ambainis [9] used to prove a $\Omega(2^n/n)$ lower bound, but with a much more careful analysis. Specifically, we observe that any Fourier Sampling algorithm would also yield an algorithm whose probability of accepting, while always small, is extremely sensitive to some specific Fourier coefficient, say $\widehat{f}(0\cdots 0)$. We then lower-bound the randomized query complexity of accepting with the required sensitivity to $\widehat{f}(0\cdots 0)$, taking advantage of the fact that $\widehat{f}(0\cdots 0)$ is simply proportional to $\sum_x f(x)$, so that all x's can be treated symmetrically. Interestingly, we also give a different, much simpler argument that yields a $\Omega(2^n/n)$ lower bound on the randomized query complexity of Fourier Fishing, which then immediately implies a $\Omega(2^n/n)$ lower bound for Fourier Sampling as well. However, if we want to improve the bound to $\Omega(2^n)$, then the original argument that Aaronson and Ambainis [9] used to prove $\Omega(2^n/n)$ seems to be needed.

In Section 7, to prove that one-way functions imply the existence of an oracle $A \in \mathsf{P/poly}$ such that $\mathsf{P}^A \neq \mathsf{BQP}^A$, we adapt a construction that was independently proposed by Zhandry [61] and by Servedio and Gortler [53]. In this construction, we first use known reductions [34, 32] to convert a one-way function into a classically-secure pseudorandom permutation, say σ. We then define a new function by $g_r(x) := \sigma(x \bmod r)$, where x is interpreted as an integer written in binary, and r is a hidden period. Finally, we argue that either Shor's

algorithm [55] leads to a quantum advantage over classical algorithms in finding the period of g_r, or else g_r was not pseudorandom, contrary to assumption. To show that subexponentially-secure one-way functions imply the existence of an oracle $A \in \mathsf{P/poly}$ relative to which Fourier Sampling is classically hard, we use similar reasoning. The main difference is that now, to construct a distinguisher against a pseudorandom function f, we need classical exponential time just to *verify* the outputs of a claimed polynomial-time classical algorithm for Fourier Sampling f – and that's why we need to assume $2^{n^{\Omega(1)}}$ security.

Finally, to prove that $\mathsf{SampBPP} = \mathsf{SampBQP}$ and $\mathsf{NP} \subseteq \mathsf{BPP}$ imply $\mathsf{SampBPP}^A = \mathsf{SampBQP}^A$ for all $A \in \mathsf{P/poly}$, we design a step-by-step classical simulation of a quantum algorithm, call it Q, that queries an oracle $A \in \mathsf{P/poly}$. We use the assumption $\mathsf{SampBPP} = \mathsf{SampBQP}$ to sample from the probability distribution over queries to A that Q makes at any given time step. Then we use the assumption $\mathsf{NP} \subseteq \mathsf{BPP}$ to guess a function $f \in \mathsf{P/poly}$ that's consistent with $n^{O(1)}$ sampled classical queries to A. Because of the limited number of functions in $\mathsf{P/poly}$, standard sample complexity bounds for PAC-learning imply that any such f that we guess will probably agree with the "true" oracle A on most inputs. Quantum-mechanical linearity then implies that the rare disagreements between f and A will have at most a small effect on the future behavior of Q.

2 Preliminaries

For a positive integer n, we use $[n]$ to denote the integers from 1 to n. Logarithms are base 2.

2.1 Quantum Circuits

We now introduce some notations for quantum circuits, which will be used throughout this paper.

In a quantum circuit, without loss of generality, we assume all gates are unitary and acting on exactly two qubits each[8].

Given a quantum circuit C, slightly abusing notation, we also use C to denote the unitary operator induced by C. Suppose there are n qubits and m gates in C; then we index the qubits from 1 to n. We also index gates from 1 to m in chronological order for convenience.

For each subset $S \subseteq [n]$ of the qubits, let $\mathcal{H}_S$ be the Hilbert space corresponding to the qubits in S, and I_S be the identity operator on $\mathcal{H}_S$. Then the unitary operator U_i for the i-th gate can be written as $U_i := O_i \otimes I_{[n]\setminus\{a_i,b_i\}}$, in which O_i is a unitary operator on $\mathcal{H}_{\{a_i,b_i\}}$ (the Hilbert space spanned by the qubits a_i and b_i), and $I_{[n]\setminus\{a_i,b_i\}}$ is the identity operator on the other qubits.

We say that a quantum circuit has depth d, if its gates can be partitioned into d layers (in chronological order), such that the gates in each layer act on disjoint pairs of qubits. Suppose the i-th layer consists of the gates in $[L_i, R_i]$. We define $C_{[r\leftarrow l]} = U_{R_r} \cdot U_{R_r-1} \ldots U_{L_l+1} \cdot U_{L_l}$, that is, the sub-circuit between the l-th layer and the r-th layer.

Base Graphs and Grids

In Sections 3 and 4, we will sometimes assume *locality* of a given quantum circuit. To formalize this notion, we define the *base graph* of a quantum circuit.

[8] Except for oracle gates, which may act on any number of qubits.

▶ **Definition 1.** Given a quantum circuit C on n qubits, its base graph $G_C = (V, E)$ is an undirected graph defined by $V = [n]$, and

$E = \{(a, b) \mid$ there is a quantum gate that acts on qubits a and $b.\}$.

We will consider a specific kind of base graph, the grids.

▶ **Definition 2.** The grid G of size $H \times W$ is a graph with vertices $V = \{(x, y) \mid x \in [H], y \in [W]\}$ and edges $E = \{(a, b) \mid |a - b|_1 = 1, a \in V, b \in V\}$, and we say that grid G has H rows and W columns.

2.2 Complexity Classes for Sampling Problems

Definitions for SampBPP and SampBQP

We adopt the following definition for sampling problems from [8].

▶ **Definition 3** (Sampling Problems, SampBPP, and SampBQP)**.** A *sampling problem* S is a collection of probability distributions $(\mathcal{D}_x)_{x\in\{0,1\}^*}$, one for each input string $x \in \{0,1\}^n$, where $\mathcal{D}_x$ is a distribution over $\{0,1\}^{p(n)}$, for some fixed polynomial p. Then SampBPP is the class of sampling problems $S = (\mathcal{D}_x)_{x\in\{0,1\}^*}$ for which there exists a probabilistic polynomial-time algorithm B that, given $\left\langle x, 0^{1/\varepsilon} \right\rangle$ as input, samples from a probability distribution $\mathcal{C}_x$ such that $\|\mathcal{C}_x - \mathcal{D}_x\| \leq \varepsilon$. SampBQP is defined the same way, except that B is a quantum algorithm rather than a classical one.

Oracle versions of these classes can also be defined in the natural way.

A Canonical Form of SampBQP Oracle Algorithms

To ease our discussion about $\mathsf{SampBQP}^{\mathcal{O}}$, we describe a canonical form of SampBQP oracle algorithms. Any other reasonable definitions of SampBQP oracle algorithms (like with quantum oracle Turing machines) can be transformed into this form easily.

Without loss of generality, we can assume a SampBQP oracle algorithm M with oracle access to $\mathcal{O}_1, oracle_2, dotsc, oracle_k$ (k is a universal constant) acts in three stages, as follows.

1. Given an input $\langle x, 0^{1/\varepsilon}\rangle$, M first uses a classical routine (which does not use the oracles) to output a quantum circuit C with $p(n, 1/\varepsilon)$ qubits and $p(n, 1/\varepsilon)$ gates in polynomial time, where p is a fixed polynomial. Note that C can use the $\mathcal{O}_1, oracle_2, dotsc, oracle_k$ gates in addition to a universal set of quantum gates.
2. Then M runs the outputted quantum circuit with the initial state $|0\rangle^{\otimes p(n,1/\varepsilon)}$, and measures all the qubits to get an outcome z in $\{0,1\}^{p(n,1/\varepsilon)}$.
3. Finally, M uses another classical routine A^{output} (which does not use the oracles) on the input z, to output its final sample $A^{\mathsf{output}}(z) \in \{0,1\}^*$.

Clearly, M solves different sampling problems (or does not solve any sampling problem at all) given different oracles $\mathcal{O}_1, oracle_2, dotsc, oracle_k$. Therefore, we use $M^{\mathcal{O}_1, oracle_2, dotsc, oracle_k}$ to indicate the particular algorithm when the oracles are $\mathcal{O}_1, oracle_2, dotsc, oracle_k$.

2.3 Distinguishing Two Pure Quantum States

We also need a standard result for distinguishing two pure quantum states.

▶ **Theorem 4** (Helstrom's decoder for two pure states). *The maximum success probability for distinguishing two pure quantum states $|\varphi_0\rangle$ and $|\varphi_1\rangle$ given with prior probabilities π_0 and π_1, is given by*

$$p_{succ} = \frac{1+\sqrt{1-4\pi_0\pi_1 F}}{2},$$

where $F := |\langle\varphi_0|\varphi_1\rangle|^2$ is the fidelity between the two states.

We'll also need that for two similar quantum states, the distributions induced by measuring them are close.

▶ **Corollary 5.** *Let $|\varphi_0\rangle$ and $|\varphi_1\rangle$ be two pure quantum state such that $||\varphi_0\rangle - |\varphi_1\rangle| \leq \varepsilon$. For a quantum state φ, define $\mathcal{D}(\varphi)$ be the distribution on $\{0,1\}^*$ induced by some quantum sampling procedure, we have*

$$\|\mathcal{D}(\varphi_0) - \mathcal{D}(\varphi_1)\| \leq \sqrt{2\varepsilon}.$$

Proof. Fix prior probabilities $\pi_0 = \pi_1 = \frac{1}{2}$.

Note that we have a distinguisher of $|\varphi_0\rangle$ and $|\varphi_1\rangle$ with success probability

$$\frac{1+\|\mathcal{D}(\varphi_0) - \mathcal{D}(\varphi_1)\|}{2}$$

by invoking that quantum sampling procedure.

By the assumption, $|\langle\varphi_0||\varphi_1\rangle| = |\langle\varphi_0|\cdot(|\varphi_0\rangle+(|\varphi_1\rangle-|\varphi_0\rangle))| \geq 1-\varepsilon$, hence $F = |\langle\varphi_0|\varphi_1\rangle|^2 \geq (1-\varepsilon)^2$. So we have

$$\frac{1+\|\mathcal{D}(\varphi_0) - \mathcal{D}(\varphi_1)\|}{2} \leq \frac{1+\sqrt{1-(1-\varepsilon)^2}}{2}.$$

This implies $\|\mathcal{D}(\varphi_0) - \mathcal{D}(\varphi_1)\|_1 \leq \sqrt{1-(1-\varepsilon)^2} = \sqrt{2\varepsilon-\varepsilon^2} \leq \sqrt{2\varepsilon}$. ◀

2.4 A Multiplicative Chernoff Bound

▶ **Lemma 6.** *Suppose $X_1, X_2, dotsc, X_n$ are independent random variables taking values in $[0,1]$. Let X denote their sum and let $\mu = \mathbb{E}[X]$. Then for any $\delta > 1$, we have*

$$\Pr[X \geq (1+\delta)\mu] \leq e^{-\frac{\delta\mu}{3}}.$$

▶ **Corollary 7.** *For any $0 < \tau$, suppose $X_1, X_2, dotsc, X_n$ are independent random variables taking values in $[0, tau]$. Let X denote their sum and let $\mu = \mathbb{E}[X]$. Then for any $\delta > 1$, we have*

$$\Pr[X \geq (1+\delta)\mu] \leq e^{-\frac{\delta\mu}{3\tau}}.$$

Proof. Replace each X_i by X_i/τ and apply the previous lemma. ◀

3 The Hardness of Quantum Circuit Sampling

We now discuss our random quantum circuit proposal for demonstrating quantum supremacy.

3.1 Preliminaries

We first introduce some notations. We use $\mathbb{U}(N)$ to denote the group of $N \times N$ unitary matrices, μ^N_{Haar} for the Haar measure on $\mathbb{U}(N)$, and μ^N_{rand} for the Haar measure on N-dimensional pure states.

For a pure state $|u\rangle$ on n qubits, we define $\mathsf{probList}(|u\rangle)$ to be the list consisting of 2^n numbers, $|\langle u|x\rangle|^2$ for each $x \in \{0,1\}^n$.

Given N real numbers $a_1, a_2, dotsc, a_N$, we use $\mathsf{uphalf}(a_1, a_2, dotsc, a_N)$ to denote the sum of the largest $N/2$ numbers among them, and we let

$$\mathsf{adv}(|u\rangle) = \mathsf{uphalf}(\mathsf{probList}(|u\rangle)).$$

Finally, we say that an output $z \in \{0,1\}^n$ is *heavy* for a quantum circuit C, if it is greater than the median of $\mathsf{probList}(C|0^n\rangle)$.

3.2 Random quantum circuit on grids

Recall that we assume a quantum circuit consists of only 2-qubit gates. Our random quantum circuit on grids of n qubits and m gates (assuming $m \geq n$) is generated as follows (though the basic structure of our hardness argument will not be very sensitive to details, and would also work for many other circuit ensembles):

- All the qubits are arranged as a $\sqrt{n} \times \sqrt{n}$ grid (see Definition 2), and a gate can only act on two adjacent qubits.
- For each $t \in [m]$ with $t \leq n$, we pick the t-th qubit and a random neighbor of it.[9]
- For each $t \in [m]$ with $t > n$, we pick a uniform random pair of adjacent qubits in the grid $\sqrt{n} \times \sqrt{n}$.
- Then, in either case, we set the t-th gate to be a unitary drawn from μ^4_{Haar} acting on these two qubits.

Slightly abusing notation, we use $\mu^{n,m}_{\mathsf{grid}}$ to denote both the above distribution on quantum circuits and the distribution on $\mathbb{U}(2^n)$ induced by it.

Conditional distribution ν_{grid}

For convenience, for a quantum circuit C, we abbreviate $\mathsf{adv}(C|0^n\rangle)$ as $\mathsf{adv}(C)$. Consider a simple quantum algorithm which measures $C|0^n\rangle$ in the computational basis to get an output z. Then by definition, $\mathsf{adv}(C)$ is simply the probability that z is heavy for C.

We want that, when a quantum circuit C is drawn, $\mathsf{adv}(C)$ is *large* (that is, bounded above $1/2$), and therefore the simple quantum algorithm has a substantial advantage on generating a heavy output, compared with the trivial algorithm of guessing a random string.

For convenience, we also consider the following conditional distribution $\nu^{n,m}_{\mathsf{grid}}$: it keeps drawing a circuit $C \leftarrow \mu^{n,m}_{\mathsf{grid}}$ until the sample circuit C satisfies $\mathsf{adv}(C) \geq 0.7$.

Lower bound on $\mathsf{adv}(C)$

We need to show that a circuit C drawn from $\nu^{n,m}_{\mathsf{grid}}$ has a large probability of having $\mathsf{adv}(C) \geq 0.7$. In order to show that, we give a cute and simple lemma, which states that the *expectation* of $\mathsf{adv}(C)$ is *large*. Surprisingly, its proof only makes use of the randomness introduced by the *very last gate*!

[9] The purpose here is to make sure that there is a gate on every qubit.

▶ **Lemma 8.** *For $n \geq 2$ and $m \geq n$*

$$\mathop{\mathbb{E}}_{C \leftarrow \mu_{\mathsf{grid}}^{n,m}} [\mathsf{adv}(C)] \geq \frac{5}{8}.$$

In fact, we conjecture that $\mathsf{adv}(C)$ is large with an *overwhelming* probability.

▶ **Conjecture 1.** *For $n \geq 2$ and $m \geq n^2$, and for all constants $\varepsilon > 0$,*

$$\Pr_{C \leftarrow \mu_{\mathsf{grid}}^{n,m}} \left[\mathsf{adv}(C) < \frac{1 + \ln 2}{2} - \varepsilon \right] < \exp\{-\Omega(n)\}.$$

We give some numerical simulation evidence for Conjecture 1 in Appendix E.

▶ Remark 9. Assuming Conjecture 1, in practice, one can sample from ν_{grid} by simply sampling from μ_{grid}, the uniform distribution over circuits—doing so only introduces an error probability of $\exp\{-\Omega(n)\}$.

3.3 The HOG Problem

Now we formally define the task in our quantum algorithm proposal.

▶ **Problem 2** (HOG, or Heavy Output Generation)**.** *Given a random quantum circuit C from $\nu_{\mathsf{grid}}^{n,m}$ for $m \geq n^2$, generate k binary strings $z_1, z_2, dotsc, z_k$ in $\{0,1\}^n$ such that at least a $2/3$ fraction of z_i's are heavy for C.*

The following proposition states that there is a simple quantum algorithm which solves the above problem with overwhelming probability.

▶ **Proposition 10.** *There is a quantum algorithm that succeeds at HOG with probability $1 - \exp\{-\Omega(k)\}$.*

Proof. The algorithm just simulates the circuit C with initial state $|0^n\rangle$, then measures in the computational basis k times independently to output k binary strings.

From the definition of ν_{grid}, we have $\mathsf{adv}(C) \geq 0.7 > 2/3$. So by a Chernoff bound, with probability $1 - \exp\{\Omega(k)\}$, at least a $2/3$ fraction of z_i's are heavy for C, in which case the algorithm solves HOG. ◀

3.4 Classical Hardness Assuming QUATH

We now state our classical hardness assumption.

▶ **Assumption 2** (QUATH, or the **Qua**ntum **Th**reshold assumption)**.** *There is no polynomial-time classical algorithm that takes as input a random quantum circuit $C \leftarrow \nu_{\mathsf{grid}}^{n,m}$ for $m \geq n^2$ and decides whether 0^n is heavy for C with success probability $1/2 + \Omega(2^{-n})$.*

▶ Remark 11. Note that $1/2$ is the success probability obtained by always outputting either 0 or 1. Therefore, the above assumption means that no efficient algorithm can beat the trivial algorithm even by $\Omega(2^{-n})$.

Next, we show that QUATH implies that no efficient classical algorithm can solve HOG.

▶ **Theorem 12.** *Assuming QUATH, no polynomial-time classical algorithm can solve HOG with probability at least* 0.99*.*

Proof. Suppose by contradiction that there is such a classical polynomial-time algorithm A. Using A, we will construct an algorithm to violate QUATH.

The algorithm is quite simple. Given a quantum circuit $C \leftarrow \nu_{\mathsf{grid}}^{n,m}$, we first draw a uniform random string $z \in \{0,1\}^n$. Then for each i such that $z_i = 1$, we apply a NOT gate on the i-th qubit. Note that this gate can be "absorbed" into the last gate acting on the i-th qubit in C. Hence, we still get a circuit C' with m gates. Moreover, it is easy to see that C' is distributed exactly the same as C even if conditioning on a particular z, and we have $\langle 0^n|C|0^n\rangle = \langle 0^n|C'|z\rangle$, which means that 0^n is heavy for C if and only if z is heavy for C'.

Next our algorithm runs A on circuit C' to get k outputs $z_1, dotsc, z_k$, and picks an output $z_{i^\star}$ among these k outputs uniformly at random. If $z_{i^\star} = z$, then the algorithm outputs 1; otherwise it outputs a uniform random bit.

Since A solves HOG with probability 0.99, we have that each z_k is heavy for C' with probability at least $0.99 \cdot 2/3$.

Now, since z is a uniform random string, the probability that our algorithm decides correctly whether z is heavy for C' is

$$\Pr[z = z_{i^\star}] \cdot 0.99 \cdot \frac{2}{3} + \Pr[z \neq z_{i^\star}] \cdot 1/2 = 2^{-n} \cdot 0.99 \cdot \frac{2}{3} + (1 - 2^{-n}) \cdot 1/2$$
$$= \frac{1}{2} + \Omega(2^{-n}).$$

But this contradicts QUATH, so we are done. ◀

3.5 Proof for Lemma 8

We first need a simple lemma which helps us to lower bound $\mathsf{adv}(|u\rangle)$.

For a pure quantum state $|u\rangle$, define

$$\mathsf{dev}(|u\rangle) = \sum_{w \in \{0,1\}^n} \left| |\langle u|w\rangle|^2 - 2^{-n} \right|.$$

In other words, $\mathsf{dev}(|u\rangle)$ measures the *non-uniformity* of the distribution obtained by measuring $|u\rangle$ in the computational basis.

The next lemma shows that, when $\mathsf{dev}(|u\rangle)$ is large, so is $\mathsf{adv}(|u\rangle)$. Therefore, in order to establish Lemma 8, it suffices to lower-bound $\mathsf{dev}(|u\rangle)$.

▶ **Lemma 13.** *For a pure quantum state $|u\rangle$, we have*

$$\mathsf{adv}(|u\rangle) \geq \frac{1}{2} + \frac{\mathsf{dev}(u)}{4}.$$

We will also need the following technical lemma.

▶ **Lemma 14.** *Let $|u\rangle \leftarrow \mu_{\mathsf{rand}}^2$. Then*

$$\mathop{\mathbb{E}}_{|u\rangle \leftarrow \mu_{\mathsf{rand}}^2} \left[\left| |\langle u|0\rangle|^2 - |\langle u|1\rangle|^2 \right| \right] = 0.5.$$

The proofs of Lemma 13 and Lemma 14 are based on simple but tedious calculations, so we defer them to Appendix B.

Now we are ready to prove Lemma 8.

Proof of Lemma 8. Surprisingly, our proof only uses the randomness introduced by the very last gate. That is, the claim holds even if there is an adversary who fixes all the gates except for the last one.

We use I_n to denote the n-qubit identity operator.
Let $C \leftarrow \mu_{\mathsf{grid}}^{n,m}$. From Lemma 13, it suffices to show that

$$\mathop{\mathbb{E}}_{C\leftarrow\mu_{\mathsf{grid}}^{n,m}}[\mathsf{dev}(C|0^n\rangle)] \geq \frac{1}{2}.$$

Suppose the last gate $U \leftarrow \mu_{\mathsf{Haar}}^4$ acts on qubits a and b. Let the unitary corresponding to the circuit before applying the last gate be V, and $|v\rangle = V|0^n\rangle$. Now, suppose we apply another unitary U_a drawn from μ_{Haar}^2 on the qubit a. It is not hard to see that U and $(U_a \otimes I_1) \cdot U$ are identically distributed. So it suffices to show that

$$\mathop{\mathbb{E}}_{U\leftarrow\mu_{\mathsf{Haar}}^4, U_a\leftarrow\mu_{\mathsf{Haar}}^2}\Big[\mathsf{adv}\Big((U_a \otimes I_{n-1})(U \otimes I_{n-2})|v\rangle\Big)\Big] \geq 0.6.$$

We are going to show that the above holds even for a fixed U. That is, fix a $U \in \mathbb{U}(4)$ and let $|u\rangle = U \otimes I_{n-2})|v\rangle$. Then we will prove that

$$\mathop{\mathbb{E}}_{U_a\leftarrow\mu_{\mathsf{Haar}}^2}\Big[\mathsf{dev}\Big((U_a \otimes I_{n-1})|v\rangle\Big)\Big] \geq \frac{1}{2}.$$

Without loss of generality, we can assume that a is the last qubit. Then we write

$$|u\rangle = \sum_{w\in\{0,1\}^n} a_w|w\rangle,$$

and

$$|z\rangle = (U_a \otimes I_{n-1})|u\rangle.$$

Now we partition the 2^n basis states into 2^{n-1} buckets, one for each string in $\{0,1\}^{n-1}$. That is, for each $p \in \{0,1\}^{n-1}$, there is a bucket that consists of basis states $\{|p0\rangle, spzp1\}$. Note that since U_a acts on the last qubit, only amplitudes of basis states in the same bucket can affect each other.

For a given $p \in \{0,1\}^{n-1}$, if both a_{p0} and a_{p1} are zero, we simply ignore this bucket. Otherwise, we can define a quantum state

$$|t_p\rangle = \frac{a_{p0}|0\rangle + a_{p1}|1\rangle}{\sqrt{|a_{p0}|^2 + |a_{p1}|^2}},$$

and

$$|z_p\rangle = U_a|t_p\rangle.$$

Clearly, we have $\langle z|p0\rangle = \sqrt{|a_{p0}|^2 + |a_{p1}|^2} \cdot \langle z_p|0\rangle$ and $\langle z|p1\rangle = \sqrt{|a_{p0}|^2 + |a_{p1}|^2} \cdot \langle z_p|1\rangle$. Plugging in, we have

$$\begin{aligned}
&\mathop{\mathbb{E}}_{U_a\leftarrow\mu_{\mathsf{Haar}}^2}\Big[\Big||\langle z|p0\rangle|^2 - 2^{-n}\Big| + \Big||\langle z|p1\rangle|^2 - 2^{-n}\Big|\Big] \\
\geq &\mathop{\mathbb{E}}_{U_a\leftarrow\mu_{\mathsf{Haar}}^2}\Big[\Big||\langle z|p0\rangle|^2 - |\langle z|p1\rangle|^2\Big|\Big] && \text{(triangle inequality)} \\
= &\big(|a_{p0}|^2 + |a_{p1}|^2\big) \cdot \mathop{\mathbb{E}}_{U_a\leftarrow\mu_{\mathsf{Haar}}^2}\Big[\Big||\langle z_p|0\rangle|^2 - |\langle z_p|1\rangle|^2\Big|\Big].
\end{aligned}$$

Now, since $|t_p\rangle$ is a pure state, and U_a is drawn from μ^2_{Haar}, we see that $|z_p\rangle$ is distributed as a Haar-random pure state. So from Lemma 14, we have

$$\mathop{\mathbb{E}}_{U_a \leftarrow \mu^2_{\mathsf{Haar}}} \left[\left| |\langle z_p|0\rangle|^2 - |\langle z_p|1\rangle|^2 \right| \right] = 0.5.$$

Therefore,

$$\mathop{\mathbb{E}}_{U_a \leftarrow \mu^2_{\mathsf{Haar}}} \left[\left| |\langle z|p0\rangle|^2 - 2^{-n} \right| + \left| |\langle z|p1\rangle|^2 - 2^{-n} \right| \right] \geq \frac{1}{2} \cdot \left(|a_{p0}|^2 + |a_{p1}|^2 \right).$$

Summing up for each $p \in \{0,1\}^{n-1}$, we have

$$\mathop{\mathbb{E}}_{U_a \leftarrow \mu^2_{\mathsf{Haar}}} [\mathsf{dev}(|z\rangle)] \geq \frac{1}{2},$$

which completes the proof. ◀

4 New Algorithms to Simulate Quantum Circuits

In this section, we present two algorithms for simulating a quantum circuit with n qubits and m gates: one algorithm for arbitrary circuits, and another for circuits that act locally on grids. What's new about these algorithms is that they use both polynomial space and close to $\exp(n)$ time (but despite that, they don't violate the QUATH assumption from Section 3, for the reason pointed out in Section 1.3). Previously, it was known how to simulate a quantum circuit in polynomial space and $\exp(m)$ time (as in the proof of $\mathsf{BQP} \subseteq \mathsf{P}^{\#\mathsf{P}}$), or in exponential space and $\exp(n)$ time.

In addition, we provide a time-space trade-off scheme, which enables even faster simulation at the cost of more space usage. See Section 2.1 for the quantum circuit notations that are used throughout this section.

4.1 Polynomial-Space Simulation Algorithms for General Quantum Circuits

We first present a simple recursive algorithm for general circuits.

▶ **Theorem 15.** *Given a quantum circuit C on n qubits with depth d, and two computational basis states $|x\rangle, spzy$, we can compute $\langle y|C|x\rangle$ in $O(n \cdot (2d)^{n+1})$ time and $O(n \log d)$ space.*

Proof. In the base case $d = 1$, the answer can be trivially computed in $O(n)$ time.

When $d > 1$, we have

$$\begin{aligned}
\langle y|C|x\rangle &= \langle y|C_{[d\leftarrow d/2+1]} \cdot C_{[d/2\leftarrow 1]}|x\rangle \\
&= \langle y|C_{[d\leftarrow d/2+1]} \left(\sum_{z\in\{0,1\}^n} |z\rangle\langle z| \right) C_{[d/2\leftarrow 1]}|x\rangle \\
&= \sum_{z\in\{0,1\}^n} \langle y|C_{[d\leftarrow d/2+1]}|z\rangle \cdot \langle z|C_{[d/2\leftarrow 1]}|x\rangle. \qquad (2)
\end{aligned}$$

Then, for each z, we calculate $\langle y|C_{[d\leftarrow d/2+1]}|z\rangle \cdot \langle z|C_{[d/2\leftarrow 1]}|x\rangle$ by recursively calling the algorithm on the two sub-circuits $C_{[d\leftarrow d/2+1]}$ and $C_{[d/2\leftarrow 1]}$ respectively; and sum them up to calculate (2).

It is easy to see the above algorithm is correct, and its running time can be analyzed as follows: let $F(d)$ be its running time on a circuit of d layers; then we have $F(1) = O(n)$, and by the above discussion

$$F(d) \leq 2^{n+1} \cdot F(\lceil d/2 \rceil) = O(n \cdot 2^{(n+1)\lceil \log d \rceil}) = O(n \cdot (2^{\lceil \log d \rceil})^{n+1}) \leq O(n \cdot (2d)^{n+1}),$$

which proves our running time bound.

Finally, we can see in each recursion level, we need $O(n)$ space to save the indices of $|x\rangle$ and $|y\rangle$, and $O(1)$ space to store an intermediate answer. Since there are at most $O(\log d)$ recursion levels, the total space is bounded by $O(n \log d)$. ◀

4.2 Faster Polynomial Space Simulation Algorithms for Grid Quantum Circuits

When a quantum circuit is spatially local, i.e., its base graph can be embedded on a grid, we can further speed up the simulation with a more sophisticated algorithm.

We first introduce a simple lemma which shows that we can find a small balanced cut in a two-dimensional grid.

▶ **Lemma 16.** *Given a grid $G = (V, E)$ of size $H \times W$ such that $|V| \geq 2$, we can find a subset $S \subset E$ such that*

- *$|S| \leq O(\sqrt{|V|})$, and*
- *after S is removed, G becomes a union of two disconnected grids with size smaller than $\frac{2}{3}|V|$.*

Proof. We can assume $H \geq W$ without loss of generality and simply set S to be the set of all the edges between the $\lfloor H/2 \rfloor$-th row and the $\lfloor H/2 \rfloor + 1$-th row; then both claims are easy to verify. ◀

We now present a faster algorithm for simulating quantum circuits on grids.

▶ **Theorem 17.** *Given a quantum circuit C on n qubits with depth d, and two computational basis states $|x\rangle, spzy$, assuming that G_C can be embedded into a two-dimensional grid with size n (with the embedding explicitly specified), we can compute $\langle y|C|x\rangle$ in $2^{O(d\sqrt{n})}$ time and $O(d \cdot n \log n)$ space.*

Proof. For ease of presentation, we slightly generalize the definition of quantum circuits: now each gate can be of the form $O_i \otimes I_{[n]\setminus\{a_i,b_i\}}$ (a 2-qubit gate) or $O_i \otimes I_{[n]\setminus\{a_i\}}$ (a 1-qubit gate) or simply $I_{[n]}$ (a 0-qubit gate, which is introduced just for convenience).

The algorithm works by trying to break the current large instance into many small instances which we then solve recursively. But unlike the algorithm in Theorem 15, which reduces an instance to many sub-instances with fewer *gates*, our algorithm here reduces an instance to many sub-instances with fewer *qubits*.

The base case, $n = 1$ qubit. In this case, all the gates are either 1-qubit or 0-qubit; hence the answer can be calculated straightforwardly in $O(m)$ time and constant space.

Cutting the grid by a small set. When $n \geq 2$, by Lemma 16, we can find a subset S of edges with $|S| \leq O(\sqrt{n})$. After S is removed, the grid becomes a union of two disconnected grids A and B (we use A, B to denote both the grids and the sets of the vertices in the grid for simplicity) with size smaller than $\frac{2}{3}n$.

Let

$$\{R = i \mid U_i \text{ is of the form } O_i \otimes I_{[n]\setminus\{a_i,b_i\}} \text{ and } (a_i, b_i) \in S\},$$

that is, the set of the indices of the gates crossing the cut S. Without loss of generality, we can assume that for each $i \in R$, we have $a_i \in A$ and $b_i \in B$.

Since in a single layer, there is at most one gate acting on a particular adjacent pair of qubits, we have

$$|R| \leq O(d\sqrt{n}).$$

Breaking the gates in R. Now, for each $i \in R$, we decompose O_i (which can be viewed as a matrix in $\mathbb{C}^{4\times 4}$) into a sum of 16 single-entry matrices $O_{i,1}, O_{i,2}, dotsc, O_{i,16}$.

Write O_i as

$$O_i = \sum_{x,y\in\{0,1\}^2} \langle y|O_i|x\rangle \cdot |y\rangle\langle x|.$$

Then we set $O_{i,j} = \langle y_j|O_i|x_j\rangle \cdot |y_j\rangle\langle x_j|$ for each $j \in [16]$, where (x_j, y_j) is the j-th ordered pair in $\{0,1\}^2 \times \{0,1\}^2$.

Decomposing the instance. Now, we are going to expand each $U_i = O_i \otimes I_{[n]\setminus\{a_i,b_i\}}$ as a sum

$$U_i = \sum_{j=1}^{16} O_{i,j} \otimes I_{[n]\setminus\{a_i,b_i\}}$$

for each $i \in R$, and therefore decompose the answer $\langle y|C|x\rangle = \langle y|U_m U_{m-1} \cdots U_1|x\rangle$ into a sum of $16^{|R|}$ terms. More concretely, for a mapping τ from R to $[16]$ and an index $i \in [m]$, we define

$$U_{i,tau} = \begin{cases} O_{i,tau(i)} \times I_{[n]\setminus\{a_i,b_i\}} & i \in R. \\ U_i & i \notin R. \end{cases}$$

Let $\mathcal{T}$ be the set of all mappings from R to $[16]$. Then we have

$$\langle y|C|x\rangle = \langle y|U_m U_{m-1} \cdots U_1|x\rangle = \sum_{\tau\in\mathcal{T}} \langle y|U_{m,tau} U_{m-1,tau} \cdots U_{1,tau}|x\rangle.$$

Dealing with the sub-instance. For each $\tau \in \mathcal{T}$ and an index $i \in [m]$, we are going to show that $U_{i,tau}$ can be decomposed as $U^A_{i,tau} \otimes U^B_{i,tau}$, where $U^A_{i,tau}$ and $U^B_{i,tau}$ are operators on $\mathcal{H}_A$ and $\mathcal{H}_B$ respectively.

When $i \in R$, by definition, there exist $x, y \in \{0,1\}^2$ and $\alpha \in \mathbb{C}$ such that

$$U_{i,tau} = \alpha \cdot |y\rangle\langle x| \otimes I_{[n]\setminus\{a_i,b_i\}} = \alpha \cdot \left(|y_0\rangle\langle x_0| \otimes I_{A\setminus\{a_i\}}\right) \otimes \left(|y_1\rangle\langle x_1| \otimes I_{B\setminus\{b_i\}}\right).$$

Otherwise $i \notin R$. In this case, if O_i is of the form $O_i \otimes I_{[n]\setminus\{a_i,b_i\}}$, then a_i, b_i must be both in A or in B and the claim trivially holds; and the claim is also obvious when O_i is of the form $O_i \otimes I_{[n]\setminus\{a_i\}}$ or $I_{[n]}$.

Moreover, one can easily verify that each $U^A_{i,tau}$ is of the form $O^A_i \otimes I_{A\setminus\{a_i,b_i\}}$ or $O^A_i \otimes I_{A\setminus\{a_i\}}$ or simply I_A, in which O^A_i is (respectively) a 2-qubit operator on $\mathcal{H}_{\{a_i,b_i\}}$ or a 1-qubit operator on $\mathcal{H}_{\{a_i\}}$), and the same holds for each $U^B_{i,tau}$.

Hence, we have

$$\begin{aligned}
&\langle y|U_m U_{m-1}\cdots U_1|x\rangle \\
&= \sum_{\tau\in\mathcal{T}} \langle y|U_{m,tau}U_{m-1,tau}\cdots U_{1,tau}|x\rangle .notag \qquad (3)\\
&= \sum_{\tau\in\mathcal{T}} \langle y|(U^A_{m,tau}\otimes U^B_{m,tau})(U^A_{m-1,tau}\otimes U^B_{m-1,tau})\cdots(U^A_{1,tau}\otimes U^B_{1,tau})|x\rangle .notag \qquad (4)\\
&= \sum_{\tau\in\mathcal{T}} \langle y_A|U^A_{m,tau}U^A_{m-1,tau}\cdots U^A_{1,tau}|x_A\rangle \cdot \langle y_B|U^B_{m,tau}U^B_{m-1,tau}\cdots U^B_{1,tau}|x_B\rangle, \qquad (5)
\end{aligned}$$

where x_A, x_B (y_A, y_B) is the projection of x (y) on $\mathcal{H}_A$ and $\mathcal{H}_B$.

So from the above discussion, we can then calculate $\langle y_A|U^A_{m,tau}U^A_{m-1,tau}\cdots U^A_{1,tau}|x_A\rangle$ with a recursive call with computational basis states $|x_A\rangle$ and $|y_A\rangle$, grid A, and m gates $U^A_{1,tau}, U^A_{2,tau}, dotsc, U^A_{m,tau}$.

The matrix element $\langle y_B|U^B_{m,tau}U^B_{m-1,tau}\cdots U^B_{1,tau}|x_B\rangle$ can be computed similarly. After that we sum up all the terms in (5) to get the answer.

Complexity analysis. Now we are going to bound the running time. Let $F(n)$ be an upper bound on the running time when the size of the remaining grid is n. Then we have

$$F(n) = \begin{cases} O(m) & \text{when } n = 1. \\ 2^{O(d\sqrt{n})} \cdot \max_{k\in[n/3,2n/3]} F(k) & \text{otherwise.} \end{cases}$$

The second case is due to the fact that the sizes of sub-instances (i.e., the sizes of A and B) lie in $[n/3, 2n/3]$, and $\mathcal{T} = 16^{|R|} = 2^{O(d\sqrt{n})}$. It is not hard to see that $F(n)$ is an increasing function, so we have $F(n) = 2^{O(d\sqrt{n})}F(2n/3)$ for $n > 1$, which further simplifies to $F(n) = 2^{O(d\sqrt{n})}$.

Finally, we can see that at each recursion level, we need $O(d \cdot n)$ space to store the circuit, and $O(1)$ space to store the intermediate answer. Since there are at most $\log n$ recursion levels, the space complexity is $O(d \cdot n \log n)$. ◀

Interestingly, by using tensor network methods, Markov and Shi [43] gave an algorithm for simulating quantum circuits on grids with similar running time to ours. However, the difference is that Markov and Shi's algorithm requires $2^{O(d\sqrt{n})}$ time *and* $2^{O(d\sqrt{n})}$ space, whereas ours requires $2^{O(d\sqrt{n})}$ time and only polynomial space.

The algorithm of Theorem 17 achieves a speedup over Theorem 15 only for small d, but we can combine it with the algorithm in Theorem 15 to get a faster algorithm for the whole range of d.

▶ **Theorem 18.** *There is a constant c such that, given a quantum circuit C on n qubits with depth d, and two computational basis states $|x\rangle, spzy$, assuming that G_C can be embedded into a two dimensional grid with size n (with the embedding explicitly specified), we can compute $\langle y|C|x\rangle$ in*

$$O(2^n \cdot \left[1 + \left(\frac{d}{c\sqrt{n}}\right)^{n+1}\right])$$

time and $O(d \cdot n \log n)$ space.

Proof. By Theorem 17, there is a constant c such that we have an $O(2^n)$ time and polynomial space algorithm for calculating $\langle y|C|x\rangle$ when the depth is at most $c\sqrt{n}$ for circuit on grids.

So we can use the same algorithm as in Theorem 15, except that we revert to the algorithm in Theorem 17 when the depth is no more than $c\sqrt{n}$.

We still let $F(d)$ be the running time on a circuit of d layers. We then have $F(d) = O(2^n)$ when $d \leq c\sqrt{n}$. From the above discussion, we can see that for $d > c\sqrt{n}$,

$$F(d) \leq 2^{n+1} \cdot F(\lceil \frac{d}{2} \rceil) = O(2^n \cdot 2^{(n+1)\lceil \log(d/c\sqrt{n}) \rceil}) = O(2^n \cdot \left(\frac{d}{c\sqrt{n}}\right)^{n+1}),$$

which proves the running time bound. And it is not hard to see that the algorithm's space usage is dominated by $O(d \cdot n \log n)$. ◀

4.3 Space-Time Trade-off Schemes

We now show how to optimize the running time for whatever space is available.

▶ **Theorem 19.** *Given a quantum circuit C on n qubits with depth d, two computational basis states $|x\rangle, spzy$ and an integer k, we can compute $\langle y|C|x\rangle$ in*

$$O(n2^{n-k} \cdot 2^{(k+1)\lceil \log d \rceil}) \leq O(n2^{n-k} \cdot (2d)^{k+1})$$

time and $O(2^{n-k} \log d)$ space.

Proof.

Decomposing the whole Hilbert space $\mathcal{H}_{[n]}$. We first decompose $\mathcal{H}_{[n]}$ into a direct sum of many subspaces. Let w_i be the i-th string in $\{0,1\}^k$ in lexicographic order. For each $i \in [2^k]$, let $\mathcal{H}_i = \mathsf{Span}(|w_i 0^{n-k}\rangle, ...|w_i 1^{n-k}\rangle)$. Then we have

$$\mathcal{H}_{[n]} = \bigoplus_{i=1}^{2^k} \mathcal{H}_i.$$

Also, let $\mathcal{P}_i$ be the projection from $\mathcal{H}_{[n]}$ to $\mathcal{H}_i$; then

$$I_{[n]} = \sum_{i=1}^{2^k} \mathcal{P}_i.$$

Now we generalize the original problem as follows: given two indices $s, t \in [2^k]$ and a pure state $|u\rangle$ in $\mathcal{H}_s$, we want to compute $\mathcal{P}_t C|u\rangle$. By choosing s and t such that $\mathcal{H}_s$ contains $|x\rangle$ and $\mathcal{H}_t$ contains $|y\rangle$, we can easily solve the original problem.

The base case $d = 1$. When there is only one layer, $\mathcal{P}_t C|u\rangle$ can be calculated straightforwardly in $O(n \cdot 2^{n-k})$ time and $O(2^{n-k})$ space.

Recursion. When $d > 1$, we have

$$\begin{aligned}
\mathcal{P}_t C|u\rangle &= \mathcal{P}_t C_{[d \leftarrow d/2+1]} \cdot C_{[d/2 \leftarrow 1]}|u\rangle \\
&= \mathcal{P}_t C_{[d \leftarrow d/2+1]} \left(\sum_{z \in [2^k]} \mathcal{P}_z \right) C_{[d/2 \leftarrow 1]}|u\rangle \\
&= \sum_{z \in [2^k]} \mathcal{P}_t C_{[d \leftarrow d/2+1]} \mathcal{P}_z C_{[d/2 \leftarrow 1]}|u\rangle.
\end{aligned}$$

We can then calculate $\mathcal{P}_t C_{[d \leftarrow d/2+1]} \mathcal{P}_z C_{[d/2 \leftarrow 1]}|u\rangle$ for each z as follows: we first use a recursive call to get $|b\rangle = \mathcal{P}_z C_{[d/2 \leftarrow 1]}|u\rangle$ and a second recursive call to compute $\mathcal{P}_t C_{[d \leftarrow d/2+1]}|b\rangle$ (note that $|b\rangle \in \mathcal{H}_z$).

Complexity analysis. It is easy to see that the total space usage is $O(2^{n-k}\log d)$, since for each i, storing a vector in $\mathcal{H}_i$ takes $O(2^{n-k})$ space, and we only need to record $O(1)$ such vectors at each recursion level. In addition, when $d=1$, we need only $O(2^{n-k})$ space.

For the running time bound, let $F(d)$ denote the running time on a circuit of d layers; then $F(1)=O(n2^{n-k})$. From the above discussion, it follows that

$$F(d)\leq 2^{k+1}\cdot F(\lceil d/2\rceil)=O(n2^{n-k}\cdot 2^{(k+1)\lceil\log(d)\rceil})=O(n2^{n-k}\cdot(2d)^{k+1}).$$ ◀

The above trade-off scheme can be further improved for quantum circuits on grids.

▶ **Theorem 20.** *There is a constant c such that, given a quantum circuit C on n qubits with depth d, two computational basis states $|x\rangle, spzy$ and an integer k, assuming that G_C can be embedded into a two dimensional grid with size n, we can compute $\langle y|C|x\rangle$ in*

$$2^{O(n)}\cdot\left[1+(2d/c\sqrt{n})^{k+1}\right]$$

time and

$$O\big(2^{n-k}\max(1,log(d/\sqrt{n}))\big)$$

space.

Proof. By Theorem 17, there is a constant c such that we have an $O(2^n)$ time algorithm for calculating $\langle y|C|x\rangle$ for circuits on grids with depth at most $c\sqrt{n}$.

Then we use the same algorithm as in Theorem 19, with the only modification that when $d\leq c\sqrt{n}$, we calculate $\mathcal{P}_t\cdot C|u\rangle$ by $2^{2(n-k)}$ calls of the algorithm in Theorem 17.

With the same analysis as in Theorem 19, when $d>c\sqrt{n}$, we can see that the total space usage is $O(2^{n-k}\log(d/c\sqrt{n}))$, and the running time is

$$O(2^{n+2(n-k)+(k+1)\lceil\log(d/c\sqrt{n})\rceil})=O(2^{O(n)}\cdot(2d/c\sqrt{n}))^{k+1}).$$

Combining with the algorithm for $d\leq c\sqrt{n}$ proves our running time and space bound. ◀

5 Strong Quantum Supremacy Theorems Must Be Non-Relativizing

In this section we show that there is an oracle relative to which SampBPP = SampBQP, yet $\mathsf{PH}^{\mathcal{O}}$ is infinite.

Recall that an oracle $\mathcal{O}$ is a function $\mathcal{O}:\{0,1\}^*\to\{0,1\}$, and the combination of two oracles $\mathcal{O}_0, oracle_1$, denoted as $\mathcal{O}_0\oplus\mathcal{O}_1$, simply maps $z\in\{0,1\}^*$ to $\mathcal{O}_{z_1}(z_2,z_3,dotsc,z_{|z|})$ (cf. citefenner2003oracle). We use $\mathcal{O}_n$ to denote the restriction of $\mathcal{O}$ on $\{0,1\}^n$.

5.1 Intuition

We have two simultaneous objectives: (1) we need SampBPP and SampBQP to be equal; and (2) we also need PH to be infinite. So it will be helpful to review some previous results on (1) and (2) separately.

- An oracle $\mathcal{O}$ such that $\mathsf{SampBPP}^{\mathcal{O}}=\mathsf{SampBQP}^{\mathcal{O}}$: in order to make two classes equal, we can use the standard method: *adding a much more powerful oracle* [16]. That is, we set $\mathcal{O}$ to be a PSPACE-complete language, like TQBF. Then it is easy to see both $\mathsf{SampBPP}^{\mathsf{TQBF}}$ and $\mathsf{SampBQP}^{\mathsf{TQBF}}$ become SampPSPACE (i.e., the class of approximate sampling problems solvable in polynomial space).
- An oracle $\mathcal{O}$ such that $\mathsf{PH}^{\mathcal{O}}$ is infinite: a line of works by Yao [60], Håstad [35], and others constructed relativized worlds where PH is infinite, and a very recent breakthrough by Rossman, Servedio, and Tan [49] even shows that PH is infinite relative to a random oracle with probability 1.

A Failed Attempt: Direct Combination

The first natural idea is to combine the previous two results straightforwardly by setting the oracle to be $\mathsf{TQBF} \oplus \mathcal{O}$, where $\mathcal{O}$ is a random oracle.

Alas, it is not hard to see that this does not work: while PH is still infinite, a SampBQP algorithm can perform Fourier Sampling (cf. Definition 27) on the random oracle bits, and it is known that no SampBPP algorithm can do that [9] (see also Theorem 33). Hence, in this case $\mathsf{SampBQP} \neq \mathsf{SampBPP}$.

Another Failed Attempt: Hiding a "Secret Random String" in a Secret Location

The failure of the naive approach suggests that we must somehow "hide" the random oracle bits, since if the SampBQP algorithm has access to them, then SampBPP and SampBQP will not be equal. More specifically, we want to hide a "secret random string" among the oracle bits so that:

1. a PH algorithm can find it, so that PH is still infinite, but
2. a SampBQP algorithm cannot find it, so that we can still make $\mathsf{SampBPP} = \mathsf{SampBQP}$ by attaching a TQBF oracle.

Inspired by the so-called cheat-sheet construction [10], it is natural to consider a direct hiding scheme. Imagine that the oracle bits are partitioned into two parts: one part is $\log N$ copies of the OR function on N bits, and another part is N binary strings $y_1, dotsc, y_N$, each with length N. Let $t = a_1, a_2, dotsc, a_{\log N} \in \{0,1\}^{\log N}$ be the answer to the copies of OR; we can also interpret t as an integer in $[N]$. Finally, set y_t to be a random input, while other y_i's are set to zero.

Intuitively, a PH algorithm can easily evaluate the $\log N$ copies of OR and then get access to the random string; while it is known that OR is hard for a quantum algorithm, so no quantum algorithm should be able to find the location of the random string efficiently.

Unfortunately, there is a fatal issue with the above approach: a SampBQP algorithm is also given an input $x \in \{0,1\}^n$ and it may *guess* that the input x denotes the location of the random string. That is, on some particular input, the SampBQP algorithm is "lucky" and gets access to the random string, which still makes SampBPP and SampBQP unequal.

Hiding the "Secret Random String" in a Bunch of OR's

Therefore, our final construction goes further. Instead of hiding the random string in a secret location amid the oracle bits, we hide it using a bunch of ORs. That is, suppose we want to provide N uniform random bits. Then we provide them each as an OR of N bits. In this way, a PH algorithm is still able to access the random bits, while a quantum algorithm, even if it's "lucky" with its additional input, still can't get access to these hidden random bits.

5.2 Implementation

The Distribution $\mathcal{D}_{\mathcal{O}}$ on Oracles

We first describe formally how to hide a random string inside a bunch of OR's by defining a distribution $\mathcal{D}_{\mathcal{O}}$ on oracles.

For notational convenience, our constructed oracles always map all odd-length binary strings to 0. So we can alternatively describe such an oracle $\mathcal{O}$ by a collection of functions $\{f_n\}_{n=0}^{+\infty}$, where each f_n is a function from $\{0,1\}^{2n} \to \{0,1\}$. That is, $\mathcal{O}_{2n}$ is set to be f_n for each n, while the $\mathcal{O}_{2n+1}$'s are all constant zero functions.

For each string $p \in \{0,1\}^n$, we use $B_{n,p}$ to denote the set of strings in $\{0,1\}^{2n}$ with p as a prefix. Now we first define a distribution $\mathcal{D}_n$ on functions $\{0,1\}^{2n} \to \{0,1\}$, from which a sample function f_n is generated as follows: initially, we set $f_n(x) = 0$ for all $x \in \{0,1\}^{2n}$; then for each $p \in \{0,1\}^n$, with probability 0.5, we pick an element e in $B_{n,p}$ at uniformly random and set $f_n(e) = 1$. Observe that by taking the OR of each $B_{n,p}$, we get a function $g(p) := \vee_{x \in B_{n,p}} f_n(x)$, which is a uniform random function from $\{0,1\}^n$ to $\{0,1\}$ by construction.

Finally, the $\mathcal{D}_n$'s induce a distribution $\mathcal{D}_{\mathcal{O}}$ on oracles, which generates an oracle $\mathcal{O}$ by drawing $f_n \sim \mathcal{D}_n$ independently for each integer n. That is, we set $\mathcal{O}_{2n}$ to be f_n, and $\mathcal{O}_{2n+1}$ to be $\mathbf{0}$, for each n.

Having defined the distribution $\mathcal{D}_{\mathcal{O}}$, we are ready to state our result formally.

▶ **Theorem 21.** *For an oracle $\mathcal{O}$ drawn from the distribution $\mathcal{D}_{\mathcal{O}}$, the following two statements hold with probability 1:*

- $\mathsf{SampBPP}^{\mathsf{TQBF},oracle} = \mathsf{SampBQP}^{\mathsf{TQBF},oracle}$.
- $\mathsf{PH}^{\mathsf{TQBF},oracle}$ *is infinite.*

From which our desired result follows immediately.

▶ **Corollary 22.** *There exists an oracle $\mathcal{O}' = TQBF \oplus \mathcal{O}$ such that $\mathsf{SampBPP}^{\mathcal{O}'} = \mathsf{SampBQP}^{\mathcal{O}'}$ and $\mathsf{PH}^{\mathcal{O}'}$ is infinite.*

The rest of this section is devoted to the proof of Theorem 21.

5.3 $\mathsf{SampBPP}^{\mathsf{TQBF},oracle} = \mathsf{SampBQP}^{\mathsf{TQBF},oracle}$ with Probability 1

We first describe an algorithm for simulating $\mathsf{SampBQP}^{\mathsf{TQBF},oracle}$ in $\mathsf{SampBPP}^{\mathsf{TQBF},oracle}$, thereby proving the first part of Theorem 21. In the following, we assume that all oracle algorithms are given access to two oracles, TQBF and $\mathcal{O}$.

Given a SampBQP oracle algorithm M, our central task is to give a SampBPP oracle algorithm that simulates M closely. Formally:

▶ **Lemma 23.** *For any* SampBQP *oracle algorithm M, there is a* SampBPP *oracle algorithm A such that:*

Let $\mathcal{O}$ be an oracle drawn from $\mathcal{D}_{\mathcal{O}}$, and let $\mathcal{D}^{M}_{x,varepsilon}$ and $\mathcal{D}^{A}_{x,varepsilon}$ be the distributions output by $M^{TQBF,oracle}$ and $A^{TQBF,oracle}$ respectively on input $\langle x, 0^{1/\varepsilon}\rangle$. Then with probability at least $1 - \exp\{-(2 \cdot |x| + 1/\varepsilon)\}$, we have

$$\|\mathcal{D}^{M}_{x,varepsilon} - \mathcal{D}^{A}_{x,varepsilon}\| \leq \varepsilon.$$

Before proving Lemma 23, we show it implies the first part of Theorem 21.

Proof of the first part of Theorem 21. Fix a SampBQP oracle algorithm M, and let $\mathcal{O}$ be an oracle drawn from $\mathcal{D}_{\mathcal{O}}$. We first show that with probability 1, there is a classical algorithm A_M such that

$$\|\mathcal{D}^{M}_{x,varepsilon} - \mathcal{D}^{A_M}_{x,varepsilon}\| \leq \varepsilon \text{ for all } x \in \{0,1\}^* \text{ and } \varepsilon = 2^{-k} \text{ for some integer } k. \quad (6)$$

Let A be the SampBPP algorithm guaranteed by Lemma 23. For an input $x \in \{0,1\}^*$ and an integer k, we call (x,k) a *bad pair* if $\|\mathcal{D}^{M}_{x,2^{-k}} - \mathcal{D}^{A}_{x,2^{-k}}\| > 2^{-k}$. By Lemma 23, the expected number of bad pairs is upper-bounded by

$$\sum_{n=1}^{+\infty} 2^n \cdot \sum_{k=1}^{+\infty} \exp(-(2n + 2^k)) \leq \sum_{n=1}^{+\infty} \sum_{k=1}^{+\infty} \exp(-(n+k)) \leq O(1).$$

This means that with probability 1, there are only finitely many bad pairs, so we can handle them by hardwiring their results into the algorithm A to get the algorithm A_M we want.

Since there are only countably many SampBQP oracle algorithms M, we see with probability 1, for every SampBQP oracle algorithm M, there is a classical algorithm A_M such that (6) holds. We claim that in that case, $\mathsf{SampBQP}^{\mathsf{TQBF},oracle} = \mathsf{SampBPP}^{\mathsf{TQBF},oracle}$.

Let $\mathcal{S}$ be a sampling problem in $\mathsf{SampBQP}^{\mathsf{TQBF},oracle}$. This means that there is a SampBQP oracle algorithm M, such that for all $x \in \{0,1\}^*$ and ε, we have $\|\mathcal{D}^M_{x,varepsilon} - \mathcal{S}_x\| \leq \varepsilon$. Let A_M be the corresponding SampBPP algorithm. Now consider the following algorithm A': given input $\langle x, 0^{1/\varepsilon}\rangle$, let k be the smallest integer such that $2^{-k} \leq \varepsilon/2$; then run A_M on input $\langle x, 0^{2^k}\rangle$ to get a sample from $\mathcal{D}^{A_M}_{x,2^{-k}}$.

Since

$$\begin{aligned}\|\mathcal{D}^{A'}_{x,varepsilon} - S_x\| &= \|\mathcal{D}^{A_M}_{x,2^{-k}} - S_x\| \\ &\leq \|\mathcal{D}^{M}_{x,2^{-k}} - \mathcal{D}^{A_M}_{x,2^{-k}}\| + \|\mathcal{D}^{M}_{x,2^{-k}} - \mathcal{S}_x\| \leq 2 \cdot 2^{-k} \leq \varepsilon,\end{aligned}$$

this means that A' solves $\mathcal{S}$ and $\mathcal{S} \in \mathsf{SampBPP}^{\mathsf{TQBF},oracle}$. So $\mathsf{SampBQP}^{\mathsf{TQBF},oracle} \subseteq \mathsf{SampBPP}^{\mathsf{TQBF},oracle}$ with probability 1, which completes the proof. ◀

We now prove Lemma 23, which is the most technical part of the whole section.

Proof of Lemma 23. Recall that from the canonical description in Section 2.2, there exists a fixed polynomial p, such that given input $\langle x, 0^{1/\varepsilon}\rangle$, the machine M first constructs a quantum circuit C with $N = p(|x|, 1/\varepsilon)$ qubits and N gates classically (C can contain TQBF and $\mathcal{O}$ gates). We first set up some notation.

Notation. Recall that $\mathcal{O}$ can be specified by a collection of functions $\{f_n\}_{n=0}^{+\infty}$, where each f_n maps $\{0,1\}^{2n}$ to $\{0,1\}$. Without loss of generality, we can assume that all the $\mathcal{O}$ gates act on an even number of qubits, and for each n, all the f_n gates act on the first $2n$ qubits.

For a function $f : \{0,1\}^k \to \{0,1\}$, we use U_f to denote the unitary operator mapping $|i\rangle$ to $(-1)^{f(i)}|i\rangle$ for $i \in \{0,1\}^k$.

Suppose there are T $\mathcal{O}$-gates in total, and suppose the i-th $\mathcal{O}$-gate is an f_{n_i} gate. Then the unitary operator U applied by the circuit C can be decomposed as

$$U = U_{T+1}(U_{f_{n_T}} \otimes I_{N-2n_T}) \cdots (U_{f_{n_2}} \otimes I_{N-2n_2})U_2(U_{f_{n_1}} \otimes I_{N-2n_1})U_1,$$

where the U_i's are the unitary operators corresponding to the sub-circuits which don't contain an $\mathcal{O}$ gate.

Our algorithm proceeds by replacing each $\mathcal{O}$-gate by a much simpler gate, one by one, without affecting the final quantum state too much. It then simulates the final circuit with the help of the TQBF oracle.

Replacing the t-th $\mathcal{O}$-gate. Suppose we have already replaced the first $t-1$ $\mathcal{O}$-gates. That is, for each $i \in [t-1]$, we replaced the f_{n_i} gate (the i-th $\mathcal{O}$-gate) with a g_i gate, and now we are going to replace the t-th $\mathcal{O}$-gate.

Let

$$|v\rangle = U_t(U_{g_{t-1}} \otimes I_{N-2n_{t-1}}) \cdots (U_{g_2} \otimes I_{N-2n_2})U_2(U_{g_1} \otimes I_{N-2n_1})U_1|0\rangle^{\otimes N},$$

which is the quantum state right before the t-th $\mathcal{O}$ gate in the circuit after the replacement.

For brevity, we use f to denote the function f_{n_t}, and we drop the subscript t of n_t when it is clear from context.

Analysis of incurred error. The t-th $\mathcal{O}$-gate is an f gate. If we replace it by a g gate, the change to the quantum state is

$$\|U_f \otimes I_{N-2n}|v\rangle - U_g \otimes I_{N-2n}|v\rangle\| = \|(U_f - U_g) \otimes I_{N-2n}|v\rangle\|.$$

We can analyze the above deviation by bounding its square. Let H be the Hilbert space spanned by the last $N-2n$ qubits, and let $\rho = \mathrm{Tr}_H[|v\rangle\langle v|]$. Then we have

$$\begin{aligned}
&\|((U_f - U_g) \otimes I_{N-2n})|v\rangle\|^2 \\
=&\,\mathrm{Tr}\Big[(U_f - U_g)^\dagger (U_f - U_g) \otimes I_{N-2n}|v\rangle\langle v|\Big] \\
=&\,\mathrm{Tr}\Big[(U_f - U_g)^\dagger (U_f - U_g)\rho\Big].
\end{aligned}$$

Note that

$$(U_f - U_g)^\dagger (U_f - U_g) = 4 \sum_{f(i)\neq g(i)} |i\rangle\langle i|$$

from the definition. So we can further simplify the above trace as

$$\mathrm{Tr}\Big[(U_f - U_g)^\dagger (U_f - U_g)\rho\Big] = 4 \sum_{f(i)\neq g(i)} \mathrm{Tr}[|i\rangle\langle i|\rho] = 4 \sum_{f(i)\neq g(i)} \langle i|\rho|i\rangle. \tag{7}$$

Now, ρ is a (mixed) quantum state on the first $2n$ bits, and $\langle i|\rho|i\rangle$ is the probability of seeing i when measuring ρ in the computational basis. So we can define a probability distribution Q on $\{0,1\}^{2n}$ by $Q(i) := \langle i|\rho|i\rangle$.

Using the distribution Q, the error term (7) can finally be simplified as:

$$4 \sum_{i\in\{0,1\}^{2n}} Q(i) \cdot [f(i) \neq g(i)] = 4 \cdot \Pr_{i\sim Q}[f(i) \neq g(i)], \tag{8}$$

where $[f(i) \neq g(i)]$ is the indicator function that takes value 1 when $f(i) \neq g(i)$ and 0 otherwise.

A posterior distribution $\mathcal{D}_n^{\mathsf{post}}$ on functions from $\{0,1\}^{2n} \to \{0,1\}$. Now, recall that $f = f_n$ is a function drawn from the distribution $\mathcal{D}_n$. Our goal is to replace f by another simple function g, such that with high probability, the introduced deviation (8) is small.

Note that when replacing the t-th $\mathcal{O}$ gate, we may already have previously queried some contents of f (i.e., it is not the first f_n gate in the circuit). So we need to consider the posterior distribution $\mathcal{D}_n^{\mathsf{post}}$ on functions from $\{0,1\}^{2n} \to \{0,1\}$. That is, we want a function g, such that with high probability over $f \sim \mathcal{D}_n^{\mathsf{post}}$, the error term (8) is small.

We use a function $f_{\mathsf{known}} : \{0,1\}^{2n} \to \{0,1,*\}$ to encode our knowledge: if $f(i)$ is not queried, then we set $f_{\mathsf{known}}(i) := *$; otherwise we set $f_{\mathsf{known}}(i) := f(i)$. Then $\mathcal{D}_n^{\mathsf{post}}$ is simply the distribution obtained from $\mathcal{D}_n$ by conditioning on the event that f is consistent with f_{known}.

We can now work out the posterior distribution $\mathcal{D}_n^{\mathsf{post}}$ from the definition of $\mathcal{D}_n$ and Bayes' rule.

For $f \sim \mathcal{D}_n^{\mathsf{post}}$, we can see that all the sets $B_{n,p}$ (recall that $B_{n,p}$ is the set of all strings in $\{0,1\}^{2n}$ with p as a prefix) are still independent. So we can consider each set separately.

For each $p \in \{0,1\}^n$, if there is an $x \in B_{n,p}$ such that $f_{\mathsf{known}}(x) = 1$, then by the construction of $\mathcal{D}_n$, all other elements $y \in B_{n,p}$ must satisfy $f(y) = 0$.

Otherwise, if there is no $x \in B_{n,p}$ such that $f_{\mathsf{known}}(x) = 1$, then we set $Z_p = |\{f_{\mathsf{known}}(x) = 0 \mid x \in B_{n,p}\}|$ and note that $|B_{n,p}| = 2^n$. By Bayes' rule, we see that with probability $\frac{1}{2 - Z_p \cdot 2^{-n}}$, all $y \in B_{n,p}$ satisfy $f(y) = 0$; and for each $y \in B_{n,p}$ such that $f_{\mathsf{known}}(y) = *$, with probability $\frac{2^{-n}}{2 - Z_p \cdot 2^{-n}}$, we have that y is the only element of $B_{n,p}$ that satisfies $f(y) = 1$.

Construction and Analysis of g. Our construction of g goes as follows: we first set $g(x) = f_{\mathsf{known}}(x)$ for all x such that $f_{\mathsf{known}}(x) \neq *$. Then for a parameter τ which will be specified later, we query all $x \in \{0,1\}^{2n}$ with $Q(x) \geq \tau$, and set $g(x) = f(x)$ for them. For all other positions of g, we simply set them to zero. Hence, there are at most $O(1/\tau) + W$ ones in g, where W denotes the number of ones in f_{known}.

The following three properties of g are immediate from the construction.

- $f(x) \neq g(x)$ implies $Q(x) \leq \tau$. (9)
- $g(x) = 1$ implies $f(x) = g(x)$. (10)
- For each $p \in \{0,1\}^n$, there is at most one $x \in B_{n,p}$ with $f(x) \neq g(x)$. (11)

Upper bounding the deviation (8). Now we are going to show that $\Pr_{x \sim Q}[f(x) \neq g(x)]$ is very small, with overwhelming probability over the posterior distribution $\mathcal{D}_n^{\mathsf{post}}$.

We first define 2^n random variables $\{X_p\}_{p \in \{0,1\}^n}$, where $X_p = \sum_{x \in B_{n,p}} Q(x) \cdot [f(x) \neq g(x)]$ for each $p \in \{0,1\}^n$. By the construction of $\mathcal{D}_n^{\mathsf{post}}$, we can see that all X_p's are independent. Moreover, by properties (9) and (11), there is at most one $x \in B_{n,p}$ such that $f(x) \neq g(x)$, and that x must satisfy $Q(x) \leq \tau$. Therefore $X_p \in [0, tau]$ for every p.

Let $X = \sum_{p \in \{0,1\}^n} X_p$, and $\mu = \mathbb{E}[X]$. Alternatively, we can write X as

$$X = \sum_{x \in \{0,1\}^{2n}} Q(x) \cdot [f(x) \neq g(x)],$$

so

$$\mu = \sum_{x \in \{0,1\}^{2n}} Q(x) \cdot \mathbb{E}[f(x) \neq g(x)].$$

We claim that $\mathbb{E}[f(x) \neq g(x)] \leq 2^{-n}$ for all $x \in \{0,1\}^{2n}$, and consequently $\mu \leq 2^{-n}$. Fix an $x \in \{0,1\}^{2n}$, and suppose $x \in B_{n,p}$. When $g(x) = 1$, we must have $f(x) = g(x)$ by property (10). When $g(x) = 0$, by the definition of $\mathcal{D}_n^{\mathsf{post}}$, we have $f(x) = 1$ with probability at most $\frac{2^{-n}}{2 - Z_p \cdot 2^{-n}} \leq 2^{-n}$. So $\mathbb{E}[f(i) \neq g(i)] \leq 2^{-n}$ in both cases and the claim is established.

Applying the Chernoff Bound. Set $\delta = \frac{\mu^{-1}\varepsilon^4}{32T^2}$. If $\delta \leq 1$, then we have

$$32T^2\varepsilon^{-4} \geq \mu^{-1} \geq 2^n.$$

This means that we can simply query all the positions in f_n using $2^{2n} = O(T^4 \cdot \varepsilon^{-8})$ queries, as this bound is polynomial in $|x|$ and $1/\varepsilon$ (recall that $T \leq N = p(|x|, 1/\varepsilon)$).

Hence, we can assume that $\delta > 1$. So by Corollary 7, we have

$$\Pr[X \geq 2\delta\mu] \leq \Pr[X \geq (1+\delta)\mu] \leq \exp\left\{-\frac{\delta\mu}{3\tau}\right\}.$$

Finally, we set $\tau = \dfrac{\varepsilon^4}{96T^2 \cdot (2n + \varepsilon^{-1} + \ln T)}$.

Therefore, with probability

$$1 - \exp\left\{-\frac{\delta\mu}{3\tau}\right\} = 1 - \exp(-(2n + \varepsilon^{-1} + \ln T)) = 1 - \frac{\exp(-(2n+\varepsilon^{-1})}{T},$$

we have

$$\|(U_f - U_g) \otimes I_{N-2n}|v\rangle\|^2 = 4 \cdot X \leq 8\delta\mu = \frac{\varepsilon^4}{4T^2},$$

which in turn implies

$$\|(U_f - U_g) \otimes I_{N-2n}|v\rangle\| \leq \frac{\varepsilon^2}{2T}.$$

Moreover, we can verify that g only has $O(1/\tau) + W = \mathrm{poly}(n, 1/\varepsilon)$ ones.

Analysis of the final circuit C^{final}. Suppose that at the end, for each $t \in [T]$, our algorithm has replaced the t-th $\mathcal{O}$-gate with a g_t gate, where g_t is a function from $\{0,1\}^{2n_t}$ to $\{0,1\}$. Let C^{final} be the circuit after the replacement.

Let

$$V = U_{T+1}(U_{g_T} \otimes I_{N-2n_T}) \cdots (U_{g_2} \otimes I_{N-2n_2})U_2(U_{g_1} \otimes I_{N-2n_1})U_1$$

be the unitary operator corresponding to C^{final}. Also, recall that U is the unitary operator corresponding to the original circuit C. We are going to show that $U|0\rangle^{\otimes N}$ and $V|0\rangle^{\otimes N}$, the final quantum states produced by U and V respectively, are very close.

We first define a sequence of intermediate quantum states. Let $|u_1\rangle = U_1|0\rangle^{\otimes N}$. Then for each $t > 1$, we define

$$|u_t\rangle = U_t(U_{f_{n_{t-1}}} \otimes I_{N-2n_{t-1}})|u_{t-1}\rangle.$$

That is, $|u_t\rangle$ is the quantum state immediately before applying the t-th $\mathcal{O}$-gate in the original circuit. Similarly, we let $|v_1\rangle = U_1|0\rangle^{\otimes N}$, and

$$|v_t\rangle = U_t(U_{g_{t-1}} \otimes I_{N-2n_{t-1}})|u_{t-1}\rangle$$

for each $t > 1$.

From the analysis of our algorithm, over $\mathcal{O} \sim \mathcal{D}_{\mathcal{O}}$, for each $t \in [T]$, with probability $1 - \exp(-(2n + \varepsilon^{-1}))/T$, we have

$$\|U_{f_{n_t}} \otimes I_{N-2n_t}|v_t\rangle - U_{g_t} \otimes I_{N-2n_t}|v_t\rangle\| \leq \frac{\varepsilon^2}{2T}. \tag{12}$$

So by a simple union bound, with probability at least $1 - \exp(-(2n + \varepsilon^{-1}))$, the above bound holds for all $t \in [T]$. We claim that in this case, for each $t \in [T+1]$, we have

$$\||v_t\rangle - |u_t\rangle\| \leq (t-1) \cdot \frac{\varepsilon^2}{2T}. \tag{13}$$

We prove this by induction. Clearly it is true for $t = 1$. When $t > 1$, suppose (13) holds for $t - 1$; then

$$\begin{aligned}
\||v_t\rangle - |u_t\rangle\| =& \|U_t(\mathcal{O}_{g_{t-1}} \otimes I_{N-2n_{t-1}})|v_{t-1}\rangle - U_t(f_{n_{t-1}} \otimes I_{N-2n_{t-1}})|u_{t-1}\rangle\| \\
=& \|U_{g_{t-1}} \otimes I_{N-2n_{t-1}}|v_{t-1}\rangle - U_{f_{n_{t-1}}} \otimes I_{N-2n_{t-1}}|u_{t-1}\rangle\| \\
\leq& \|U_{g_{t-1}} \otimes I_{N-2n_{t-1}}|v_{t-1}\rangle - U_{f_{n_{t-1}}} \otimes I_{N-2n_{t-1}}|v_{t-1}\rangle\| \\
&+ \|U_{f_{n_{t-1}}} \otimes I_{N-2n_{t-1}}|v_{t-1}\rangle - U_{f_{n_{t-1}}} \otimes I_{N-2n_{t-1}}|u_{t-1}\rangle\| \\
\leq& \frac{\varepsilon^2}{2T} + \||u_{t-1}\rangle - |v_{t-1}\rangle\| \leq (t-1) \cdot \frac{\varepsilon^2}{2T},
\end{aligned}$$

where the second line holds by the fact that U_t is unitary, the third line holds by the triangle inequality, and the last line holds by (12) and the induction hypothesis.

Upper-bounding the error. Therefore, with probability at least $1 - \exp(-(2n + \varepsilon^{-1}))$, we have

$$\||v_{T+1}\rangle - |u_{T+1}\rangle\| = \|U|0\rangle^{\otimes N} - V|0\rangle^{\otimes N}\| \leq \frac{\varepsilon^2}{2}.$$

Now, our classical algorithm A then simulates stage 2 and 3 of the SampBQP algorithm M straightforwardly. That is, it first takes a sample z by measuring $|v_{T+1}\rangle$ in the computational basis, and then outputs $A^{\mathsf{output}}(z)$ as its sample, where A^{output} is the classical algorithm used by M in stage 3.

From our previous analysis, A queries the oracle only $\mathrm{poly}(n, 1/\varepsilon)$ times. In addition, it is not hard to see that all the computations can be done in PSPACE, and therefore can be implemented in $\mathrm{poly}(n, 1/\varepsilon)$ time with the help of the TQBF oracle. So A is a SampBPP algorithm.

By Corollary 5, with probability at least $1-\exp(-(2n+\varepsilon^{-1}))$, the distribution $\mathcal{D}^A_{x,varepsilon}$ outputted by A satisfies

$$\|\mathcal{D}^A_{x,varepsilon} - \mathcal{D}^M_{x,varepsilon}\| \leq \sqrt{2 \cdot \frac{\varepsilon^2}{2}} = \varepsilon,$$

and this completes the proof of Lemma 23. ◀

5.4 $\mathsf{PH}^{\mathsf{TQBF},oracle}$ is Infinite with Probability 1

For the second part of Theorem 21, we resort to the well-known connection between PH and constant-depth circuit lower bounds.

The Average Case Constant-depth Circuit Lower Bound

For convenience, we will use the recent breakthrough result by Rossman, Servedio, and Tan [49], which shows that PH is infinite relative to a random oracle with probability 1. (Earlier constructions of oracles making PH infinite would also have worked for us, but a random oracle is a particularly nice choice.)

▶ **Theorem 24.** *Let $2 \leq d \leq \frac{c\sqrt{\log n}}{\log\log n}$, where $c > 0$ is an absolute constant. Let Sipser_d be the explicit n-variable read-once monotone depth-d formula described in [49]. Then any circuit C' of depth at most $d-1$ and size at most $S = 2^{n^{\frac{1}{6(d-1)}}}$ over $\{0,1\}^n$ agrees with Sipser_d on at most $(\frac{1}{2} + n^{-\Omega(1/d)}) \cdot 2^n$ inputs.*

$\mathcal{D}_n$ as a Distribution on $\{0,1\}^{2^{2n}}$

In order to use the above result to prove the second part of Theorem 21, we need to interpret $\mathcal{D}_n$ (originally a distribution over functions mapping $\{0,1\}^{2n}$ to $\{0,1\}$) as a distribution on $\{0,1\}^{2^{2n}}$ in the following way.

Let τ be the bijection between $[2^{2n}]$ and $\{0,1\}^{2n}$ that maps an integer $i \in [2^{2n}]$ to the i-th binary string in $\{0,1\}^{2n}$ in lexicographic order. Then a function $f : \{0,1\}^{2n} \to \{0,1\}$ is equivalent to a binary string $x^f \in \{0,1\}^{2^{2n}}$, where the i-th bit of x^f, denoted x^f_i, equals $f(\tau(i))$. Clearly this is a bijection between functions from $\{0,1\}^{2n}$ to $\{0,1\}$ and binary strings in $\{0,1\}^{2^{2n}}$.

For notational simplicity, when we say a binary string $x \in \{0,1\}^{2^{2n}}$ is drawn from $\mathcal{D}_n$, it means x is generated by first drawing a sample function $f \sim \mathcal{D}_n$ and then setting $x = x^f$.

Note that for $p \in \{0,1\}^n$, if p is the i-th binary string in $\{0,1\}^n$, then the set $B_{n,p}$ corresponds to the bits $x_{(i-1)2^n+1}, dotsc, x_{i2^n}$.

Distributional Constant-Depth Circuit Lower Bound over $\mathcal{D}_n$

Now we are ready to state our distributional circuit lower bound over $\mathcal{D}_n$ formally.

▶ **Lemma 25.** *For an integer n, let $N = 2^n$ and Sipser_d be the N-variable Sipser function as in Theorem 24.*

Consider the Boolean function $(\mathsf{Sipser}_d \circ \mathsf{OR})$ on $\{0,1\}^{N^2}$ defined as follows:
Given inputs $x_1, x_2, dotsc, x_{N^2}$, for each $1 \le i \le N$, set

$$z_i := \vee_{j=(i-1)N+1}^{iN} x_j,$$

and

$$(\mathsf{Sipser}_d \circ \mathsf{OR})(x) := \mathsf{Sipser}_d(z).$$

Then any circuit C' of depth at most $d-1$ and size at most $S = 2^{N^{\frac{1}{6(d-1)}}}$ over $\{0,1\}^{N^2}$ agrees with $(\mathsf{Sipser}_d \circ \mathsf{OR})$ with probability at most $\frac{1}{2} + N^{-\Omega(1/d)}$ when inputs are drawn from the distribution $\mathcal{D}_n$.

Before proving Lemma 25, we show that it implies the second part of Theorem 21 easily.

Proof of the second part of Theorem 21. Consider the function $(\mathsf{Sipser}_d \circ \mathsf{OR})$ defined as in Lemma 25. It is easy to see that it has a polynomial-size circuit (in fact, a formula) of depth $d+1$; and by Lemma 25, every polynomial size circuit of depth $d-1$ has at most $\frac{1}{2} + o(1)$ correlation with it when the inputs are drawn from the distribution $\mathcal{D}_n$. So it follows from the standard connection between PH and AC_0 that $\mathsf{PH}^{\mathcal{O}}$ is infinite with probability 1 when $\mathcal{O} \sim \mathcal{D}_{\mathcal{O}}$. ◀

Finally, we prove Lemma 25.

Proof of Lemma 25. By Theorem 24, there is a universal constant c, such that any circuit C of depth at most $d-1$ and size at most S over $\{0,1\}^N$ agrees with Sipser_d on at most $\left(\frac{1}{2} + N^{-c/d}\right) \cdot 2^N$ inputs.

We are going to show this lemma holds for the same c. Suppose not; then we have a circuit C of depth at most $d-1$ and size at most $S = 2^{N^{\frac{1}{6(d-1)}}}$ over $\{0,1\}^{N^2}$, such that

$$\Pr_{x\sim\mathcal{D}_n}[C(x) = (\mathsf{Sipser}_d \circ \mathsf{OR})(x)] > \frac{1}{2} + N^{-c/d}.$$

Now, for each $y_1, y_2, dotsc, y_N \in [N]^N$, we define a distribution $\mathcal{D}_n^{y_1,y_2,dotsc,y_N}$ on $\{0,1\}^{N^2}$ as follows. To generate a sample $x \sim \mathcal{D}_n^{y_1,y_2,dotsc,y_N}$, we first set $x = 0^{N^2}$. Then for each $i \in [N]$, we set $x_{(i-1)N+y_i}$ to 1 with probability $1/2$.

By construction, we can see for all x in the support of $\mathcal{D}_n^{y_1,y_2,dotsc,y_N}$,

$$(\mathsf{Sipser}_d \circ \mathsf{OR})(x) = \mathsf{Sipser}_d(x_{y_1}, x_{N+y_2}, x_{2N+y_3}, dotsc, x_{(N-1)N+y_N}).$$

Moreover, by definition, $\mathcal{D}_n$ is just the average of these distributions:

$$\mathcal{D}_n = N^{-N} \cdot \sum_{y_1,y_2,dotsc,y_N} \mathcal{D}_n^{y_1,y_2,dotsc,y_N}.$$

By an averaging argument, there exist $y_1, y_2, dotsc, y_N \in [N]^N$ such that

$$\Pr_{x\sim\mathcal{D}_n^{y_1,y_2,dotsc,y_N}}[C(x) = (\mathsf{Sipser}_d \circ \mathsf{OR})(x)] > \frac{1}{2} + N^{-c/d}.$$

Setting $x_{(i-1)N+y_i} = z_i$ for each i, and all other inputs to 0 in the circuit C, we then have a circuit D of size at most S and depth at most $d-1$ over $\{0,1\}^N$. And by the construction of $\mathcal{D}_n^{y_1,y_2,dotsc,y_N}$ and the definition of the function $(\mathsf{Sipser}_d \circ \mathsf{OR})$, we see that D agrees with Sipser_d on at least a $\frac{1}{2} + N^{-c/d}$ fraction of inputs. But this is a contradiction. ◀

6 Maximal Quantum Supremacy for Black-Box Sampling and Relation Problems

In this section we present our results about Fourier Fishing and Fourier Sampling.

We will establish an $\Omega(N/\log N)$ lower bound on the classical query complexity of Fourier Fishing, as well as an optimal $\Omega(N)$ lower bound on the classical query complexity of Fourier Sampling.

6.1 Preliminaries

We begin by introducing some useful notations. Throughout this section, given a function $f : \{0,1\}^n \to \{-1,1\}$, we define the Fourier coefficient

$$\widehat{f}(z) = 2^{-n/2} \sum_{x\in\{0,1\}^n} f(x) \cdot (-1)^{x\cdot z}$$

for each $z \in \{0,1\}^n$.

We also define

$$\mathsf{adv}(f) := 2^{-n} \cdot \sum_{z\in\{0,1\}^n, |\widehat{f}(z)|\geq 1} \widehat{f}(z)^2,$$

and set $N = 2^n$.

The following two constants will be used frequently in this section.

$$\mathsf{Succ}_Q = \frac{2}{\sqrt{2\pi}} \int_1^{+\infty} x^2 e^{-x^2/2} dx \approx 0.801 \text{ and } \mathsf{Succ}_R = \frac{2}{\sqrt{2\pi}} \int_1^{+\infty} e^{-x^2/2} dx \approx 0.317.$$

Finally, we use $\mathcal{U}_n$ to denote the uniform distribution on functions $f : \{0,1\}^n \to \{-1,1\}$.

An Approximate Formula for the Binomial Coefficients

We also need the following lemma to approximate the binomial coefficients to ease some calculations in our proofs.

▶ **Lemma 26** ((5.41) in [56]). *For value n and $|k - n/2| = o(n^{2/3})$, we have*

$$\binom{n}{k} \approx \binom{n}{n/2} \cdot e^{-\frac{(k-n/2)^2}{n/2}}$$

and

$$\ln \binom{n}{k} = \ln \binom{n}{n/2} - \frac{(k-n/2)^2}{n/2} + o(1).$$

6.2 Fourier Fishing and Fourier Sampling

We now formally define the Fourier Fishing and the Fourier Sampling problems.

▶ **Definition 27.** We are given oracle access to a function $f : \{0,1\}^n \to \{-1,1\}$.

In Fourier Sampling (or Fsampling in short), our task is to sample from a distribution $\mathcal{D}$ over $\{0,1\}^n$ such that $\|\mathcal{D} - \mathcal{D}_f\| \leq \varepsilon$, where $\mathcal{D}_f$ is the distribution defined by

$$\Pr_{\mathcal{D}_f}[y] = 2^{-n}\widehat{f}(y)^2 = \left(\frac{1}{2^n}\sum_{x\in\{0,1\}^n} f(x)(-1)^{x\cdot y}\right)^2.$$

In Fourier Fishing (or Ffishing in short), we want to find a z such that $|\widehat{f}(z)| \geq 1$. We also define a promise version of Fourier Fishing (promise-Ffishing for short), where the function f is promised to satisfy $\mathsf{adv}(f) \geq \mathsf{Succ}_Q - \frac{1}{n}$.

A Simple 1-Query Quantum Algorithm

Next we describe a simple 1-query quantum algorithm for both problems. It consists of a round of Hadamard gates, then a query to f, then another round of Hadamard gates, then a measurement in the computational basis.

The following lemma follows directly from the definitions of Fsampling and Ffishing.

▶ **Lemma 28.** *Given oracle access to a function $f : \{0,1\}^n \to \{-1,1\}$, the above algorithm solves Fsampling exactly (i.e. with $\varepsilon = 0$), and Ffishing with probability* $\mathsf{adv}(f)$.

We can now explain the meanings of the constants Succ_Q and Succ_R. When the function f is drawn from $\mathcal{U}_n$, by a simple calculation, we can see that Succ_Q is the success probability for the above simple quantum algorithm on Fourier Fishing, and Succ_R is the success probability for an algorithm outputting a uniform random string in $\{0,1\}^n$.

6.3 The $\Omega(N/\log N)$ Lower Bound for Fourier Fishing

We begin with the $\Omega(N/\log N)$ randomized lower bound for Fourier Fishing. Formally:

▶ **Theorem 29.** *There is no $o(N/\log N)$-query randomized algorithm that solves promise-Ffishing with* $\mathsf{Succ}_R + \Omega(1)$ *success probability.*

To prove Theorem 29, we first show that when the function f is drawn from $\mathcal{U}_n$, no classical algorithm with $o(N/\log N)$ queries can solve Ffishing with probability $\mathsf{Succ}_R + \Omega(1)$; we then show with high probability, a function $f \leftarrow \mathcal{U}_n$ satisfies the promise of promise-Ffishing. Formally, we have the following two lemmas.

▶ **Lemma 30.** *For large enough n,*

$$\Pr_{f \leftarrow \mathcal{U}_n}\left[\mathsf{adv}(f) < \mathsf{Succ}_Q - \frac{1}{n}\right] < \frac{1}{n}.$$

▶ **Lemma 31.** *Over $f \leftarrow \mathcal{U}_n$, no randomized algorithm with $o(N/\log N)$ queries can solve Ffishing with probability*

$$\mathsf{Succ}_R + \Omega(1).$$

Before proving these two technical lemmas, we show that they together imply Theorem 29 easily.

Proof of Theorem 29. Suppose by contradiction that there is an $o(N/\log N)$ query randomized algorithm A which has a $\mathsf{Succ}_R + \Omega(1)$ success probability for promise-Ffishing. From Lemma 30, a $1 - o(1)$ fraction of all functions from $\{0,1\}^n \to \{-1,1\}$ satisfy the promise of promise-Ffishing. Therefore, when the sample function f is drawn from $\mathcal{U}_f$, with probability $1 - o(1)$ it satisfies the promise of promise-Ffishing, and consequently A has a $\mathsf{Succ}_R + \Omega(1)$ success probability of solving Ffishing with that f. This means that A has a success probability of

$$(1 - o(1)) \cdot (\mathsf{Succ}_R + \Omega(1)) = \mathsf{Succ}_R + \Omega(1)$$

when $f \leftarrow \mathcal{U}_n$, contradicting Lemma 31. ◀

The proof of Lemma 30 is based on a tedious calculation so we defer it to Appendix C. Now we prove Lemma 31.

Proof of Lemma 31. By Yao's principle, it suffices to consider only deterministic algorithms, and we can assume the algorithm A makes exactly $t = o(N/\log N)$ queries without loss of generality.

Notations. Suppose that at the end of the algorithm, A has queried the entries in a subset $S \subseteq \{0,1\}^n$ such that $|S| = t$.

For each $z \in \{0,1\}^n$, we define

$$\widehat{f}_{\mathsf{seen}}(z) = \frac{1}{\sqrt{t}} \sum_{x \in S} f(x) \cdot (-1)^{x \cdot z}$$

and similarly

$$\widehat{f}_{\mathsf{unseen}} f(z) = \frac{1}{\sqrt{N-t}} \sum_{x \in \{0,1\}^n \setminus S} f(x) \cdot (-1)^{x \cdot z}.$$

From the definitions of $\widehat{f}(z)$, $\widehat{f}_{\mathsf{seen}}(z)$ and $\widehat{f}_{\mathsf{unseen}}(z)$, and note that $N/t = \omega(\log N) = \omega(\ln N)$, we have

$$\begin{aligned}\widehat{f}(z) &= \left(\sqrt{t} \cdot \widehat{f}_{\mathsf{seen}}(z) + \sqrt{N-t} \cdot \widehat{f}_{\mathsf{unseen}}(z)\right) \Big/ \sqrt{N} \\ &= \widehat{f}_{\mathsf{seen}}(z)/\omega(\sqrt{\ln N}) + \widehat{f}_{\mathsf{unseen}}(z) \cdot (1 - o(1)). \end{aligned} \tag{14}$$

W.h.p. $\widehat{f}_{\mathsf{seen}}(z)$ is small for all $z \in \{0,1\}^n$. We first show that, with probability at least $1 - o(1)$ over $f \leftarrow \mathcal{U}_n$, we have $|\widehat{f}_{\mathsf{seen}}(z)| \leq 2\sqrt{\ln N}$ for all $z \in \{0,1\}^n$.

Fix a $z \in \{0,1\}^n$, and note that for the algorithm A, even though which position to query next might depend on the history, the value in that position is a uniform random bit in $\{-1,1\}$. So $\widehat{f}_{\mathsf{seen}}(z)$ is a sum of t uniform i.i.d. random variables in $\{-1,1\}$.

Therefore, the probability that $|\widehat{f}_{\mathsf{seen}}(z)| > 2\sqrt{\ln N}$ for this fixed z is

$$\frac{2}{\sqrt{2\pi}} \int_{2\sqrt{\ln N}}^{+\infty} e^{-x^2/2} dx = o\left(\frac{1}{N}\right).$$

Then by a simple union bound, with probability $1 - o(1)$, there is no $z \in \{0,1\}^n$ such that $|\widehat{f}_{\mathsf{seen}}(z)| > 2\sqrt{\ln N}$ at the end of t queries. We denote the nonexistence of such a z as the event $\mathcal{E}_{\mathsf{bad}}$.

The lower bound. In the following we condition on $\mathcal{E}_{\mathsf{bad}}$. We show in this case, A cannot solve Ffishing with a success probability better than Succ_R, thereby proving the lower bound.

From (14), for each $z \in \{0,1\}^n$, we have

$$\widehat{f}(z) = o(1) + \widehat{f}_{\mathsf{unseen}}(z) \cdot (1 - o(1)).$$

Therefore, the probability of $|\widehat{f}(z)| \geq 1$ is bounded by the probability that $|\widehat{f}_{\mathsf{unseen}}(z)| \geq 1 - o(1)$. Since $\widehat{f}_{\mathsf{unseen}}(z)$ is independent of all the seen values in S, we have

$$\begin{aligned}
\Pr\left[\widehat{f}_{\mathsf{unseen}}(z) \geq 1 - o(1)\right] &= \frac{2}{\sqrt{2\pi}} \int_{1-o(1)}^{+\infty} e^{-x^2/2} dx \\
&= \frac{2}{\sqrt{2\pi}} \int_{1}^{+\infty} e^{-x^2/2} dx + o(1) \\
&= \mathsf{Succ}_R + o(1).
\end{aligned}$$

Hence, no matter which z is outputted by A, we have $|\widehat{f}(z)| \geq 1$ with probability at most $\mathsf{Succ}_R + o(1)$. That means that if we condition on $\mathcal{E}_{\mathsf{bad}}$, then A cannot solve Ffishing with probability $\mathsf{Succ}_R + \Omega(1)$. As $\mathcal{E}_{\mathsf{bad}}$ happens with probability $1 - o(1)$, this finishes the proof. ◀

6.4 The Optimal $\Omega(N)$ Lower Bound for Fourier Sampling

We first show that in fact, Lemma 31 already implies an $\Omega(N/\log N)$ lower bound for Fourier Sampling, which holds for a quite large ε.

▶ **Theorem 32.** *For any $\varepsilon < \mathsf{Succ}_Q - \mathsf{Succ}_R \approx 0.483$, the randomized query complexity for Fsampling is $\Omega(N/\log N)$.*

Proof. Note when $f \leftarrow \mathcal{U}_n$, an exact algorithm for Fsampling can be used to solve Ffishing with probability Succ_Q. Hence, a sampling algorithm for Fsampling with total variance $\leq \varepsilon$ can solve Ffishing with probability at least $\mathsf{Succ}_Q - \varepsilon$, when $f \leftarrow \mathcal{U}_n$.

Then the lower bound follows directly from Lemma 31. ◀

Next we prove the optimal $\Omega(N)$ lower bound for Fourier Sampling.

▶ **Theorem 33.** *There is a constant $\varepsilon > 0$, such that any randomized algorithm solving Fsampling with error at most ε needs $\Omega(N)$ queries.*

Proof.

Reduction to a simpler problem. Sampling problems are hard to approach, so we first reduce to a much simpler problem with Boolean output ("accept" or "reject").

Let A be a randomized algorithm for Fsampling with total variance $\leq \varepsilon$. For a function $f : \{0,1\}^n \to \{-1,1\}$ and $y \in \{0,1\}^n$, we set $p_{f,y}$ to be the probability that A outputs y with oracle access to f.

By the definition of Fsampling, for all f, we have

$$\frac{1}{2} \sum_{y \in \{0,1\}^n} \left| p_{f,y} - 2^{-n} \widehat{f}(y)^2 \right| \leq \varepsilon.$$

By an averaging argument, this implies that there exists a $y^* \in \{0,1\}^n$ such that

$$\mathop{\mathbb{E}}_{f \leftarrow \mathcal{U}_n} \left[\left| p_{f,y^*} - 2^{-n} \widehat{f}(y^*)^2 \right| \right] \leq \frac{2\varepsilon}{N}.$$

Then by Markov's inequality, we have

$$\left| p_{f,y^*} - 2^{-n} \widehat{f}(y^*)^2 \right| \leq \frac{400\varepsilon}{N},$$

for at least a 199/200 fraction of f's. Now we set $\varepsilon = \frac{1}{400} \cdot \frac{1}{100}$.

Without loss of generality, we can assume that $y^* = 0^n$. Let $z_i := \frac{1 + f(x_i)}{2}$ (where $x_1, x_2, dotsc, x_N$ is a lexicographic ordering of inputs), $Z := (z_1, dotsc, z_N)$ and $|Z| := \sum_{i=1}^{N} z_i$. Then we have

$$2^{-n} \widehat{f}(0^n)^2 = \left(\frac{2|Z|}{N} - 1 \right)^2.$$

Now we can simplify the question to one of how many z_i's the algorithm A needs to query, in order to output 0^n (we call it "accept" for convenience) with a probability $p_Z = p_{f,0^n}$ that satisfies

$$\left| p_Z - \left(\frac{2|Z|}{N} - 1 \right)^2 \right| \leq \frac{400\varepsilon}{N} \leq \frac{0.01}{N} \tag{15}$$

with probability at least 199/200 over $Z \in \{0,1\}^N$.

Analysis of the acceptance probability of A. Without loss of generality, we can assume that A non-adaptively queries t randomly-chosen inputs $z_{i_1}, z_{i_2}, dotsc, z_{i_t}$, and then accepts with a probability q_k that depends solely on $k := z_{i_1} + \cdots + z_{i_t}$. The reason is that we can change any other algorithm into this restricted form by averaging over all $N!$ permutations of Z without affecting its correctness.

Let p_w be the probability that A accepts when $|Z| = w$. Then

$$p_w = \sum_{k=0}^{t} q_k \cdot r_{k,w},$$

where $r_{k,w} := \binom{t}{k} \binom{N-t}{w-k} \Big/ \binom{N}{w}$, is the probability that $z_{i_1} + \cdots + z_{i_t} = k$ conditioned on $|Z| = w$.

Construction and Analysis of the sets U, V, W. Now, consider the following three sets:

$$U := \left\{ Z : \left| |Z| - \frac{N}{2} \right| \leq \frac{\sqrt{N}}{20} \right\},$$
$$V := \left\{ Z : \left(1 - \frac{1}{20}\right)\frac{\sqrt{N}}{2} \leq |Z| - \frac{N}{2} \leq \frac{\sqrt{N}}{2} \right\},$$
$$W := \left\{ Z : \left(1 - \frac{1}{20}\right)\sqrt{N} \leq |Z| - \frac{N}{2} \leq \sqrt{N} \right\}.$$

We calculate the probability that a uniform random Z belongs to these three sets. For a sufficiently large N, we have

$$\Pr_Z[Z \in U] \geq \mathrm{erf}\left(\frac{\sqrt{2}}{20}\right) - o(1) > 0.075,$$
$$\Pr_Z[Z \in V] \geq \frac{1}{2} \cdot \left(\mathrm{erf}\left(\frac{\sqrt{2}}{2}\right) - \mathrm{erf}\left(\frac{\sqrt{2}}{2} \cdot \frac{19}{20}\right)\right) - o(1) > 0.01,$$
$$\Pr_Z[Z \in W] \geq \frac{1}{2} \cdot \left(\mathrm{erf}(\sqrt{2}) - \mathrm{erf}\left(\sqrt{2} \cdot \frac{19}{20}\right)\right) - o(1) > 0.005.$$

Construction and Analysis of w_0, w_1, w_2. Since all $\Pr_Z[Z \in U], \Pr_Z[Z \in V], \Pr_Z[Z \in W] > 0.005$, and recall that for at least a $1 - 0.005$ fraction of Z, we have

$$\left| p_Z - \left(\frac{2|Z|}{N} - 1\right)^2 \right| \leq \frac{400\varepsilon}{N} \leq \frac{0.01}{N}.$$

So there must exist $w_0 \in U, w_1 \in V, w_2 \in W$ such that

$$\left| p_{w_i} - 4 \cdot \left(\frac{w_i - N/2}{N}\right)^2 \right| \leq \frac{0.01}{N} \tag{16}$$

for each $i \in \{0, 1, 2\}$.

To ease our calculation, let $u_i = \dfrac{w_i - N/2}{\sqrt{N}}$, then we have $w_i = N/2 + u_i\sqrt{N}$. By the definition of the u_i's, we also have $|u_0| \leq \dfrac{1}{20}, u_1 \in [0.475, 0.5], u_2 \in [0.95, 1]$.

Plugging in u_i's, for each $i \in \{0, 1, 2\}$, equation (16) simplifies to

$$\left| p_{w_i} - \frac{4u_i^2}{N} \right| \leq \frac{0.01}{N}. \tag{17}$$

We can calculate the ranges of the p_{w_i}'s by plugging the ranges of the u_i's,

$$p_{w_0} \leq \frac{0.02}{N},$$
$$p_{w_1} \in \left[\frac{0.95^2 - 0.01}{N}, frac1 + 0.01N\right] \subseteq \left[\frac{0.89}{N}, frac1.01N\right],$$
$$p_{w_2} \in \left[\frac{4 \cdot 0.95^2 - 0.01}{N}, frac4 + 0.01N\right] \subseteq \left[\frac{3.6}{N}, frac4.01N\right].$$

We are going to show that the above is impossible when $t = o(N)$. That is, one cannot set the q_k's in such a way that all p_{w_i}'s satisfy the above constraints when $t = o(N)$.

It is safe to set q_k to zero when $|k-t/2|$ is large. To simplify the matters, we first show that we can set nearly all the q_k's to zero. By the Chernoff bound without replacement, for each w_i and large enough c we have

$$\begin{aligned}
&\sum_{k:|k-t/2|>c\sqrt{t}} r_{k,w_i} \\
&= \Pr\Big[\big|z_{i_1}+\cdots+z_{i_t}-\tfrac{t}{2}\big| \geq c\sqrt{t} : |Z| = w_i = \tfrac{N}{2}+u_i\sqrt{N}\Big] \\
&\leq \Pr\Big[\Big|z_{i_1}+\cdots+z_{i_t}-\Big(\tfrac{t}{2}+\tfrac{u_i t}{\sqrt{N}}\Big)\Big| \geq c\sqrt{t}-\Big|\Big(\tfrac{t}{2}+\tfrac{u_i t}{\sqrt{N}}\Big)-\tfrac{t}{2}\Big| : |Z| = w_i = \tfrac{N}{2}+u_i\sqrt{N}\Big] \\
&\leq \Pr\Big[\Big|z_{i_1}+\cdots+z_{i_t}-\Big(\tfrac{t}{2}+\tfrac{u_i t}{\sqrt{N}}\Big)\Big| \geq c\sqrt{t}-|u_i|\sqrt{t} : |Z| = \tfrac{N}{2}+u_i\sqrt{N}\Big] \qquad (\tfrac{t}{\sqrt{N}} \leq \sqrt{t}) \\
&\leq \exp\Big\{-2\tfrac{(c\sqrt{t}-|u_i|\sqrt{t})^2}{t}\Big\} \\
&= \exp\big\{-2(c-|u_i|)^2\big\} \\
&\leq \exp\big\{\Omega(c^2)\big\}.
\end{aligned}$$

Then we can set $c = c_1\sqrt{\ln N}$ for a sufficiently large constant c_1, so that for all w_i's,

$$\sum_{k:|k-t/2|>c\sqrt{t}} r_{k,w_i} \leq \frac{1}{N^2}.$$

This means that we can simply set all q_k's with $|k-t/2| > c\sqrt{t} = c_1\sqrt{t\ln N}$ to zero, and only consider k such that $|k-t/2| \leq c_1\sqrt{t\ln N}$, as this only changes each p_{w_i} by a negligible value. From now on, we call an integer k *valid*, if $|k-t/2| \leq c_1\sqrt{t\ln N}$.

Either $\frac{r_{k,w_0}}{r_{k,w_1}} \geq 0.05$ or $\frac{r_{k,w_2}}{r_{k,w_1}} \geq 10$. Now, we are going to show the most technical part of this proof: for all valid k, we have either

$$\frac{r_{k,w_0}}{r_{k,w_1}} \geq 0.05 \text{ or } \frac{r_{k,w_2}}{r_{k,w_1}} \geq 10. \tag{18}$$

Suppose for contradiction that there is a valid k that satisfies

$$\frac{r_{k,w_0}}{r_{k,w_1}} < 0.05 \text{ and } \frac{r_{k,w_2}}{r_{k,w_1}} < 10. \tag{19}$$

Estimation of r_{k,w_i}'s. We first use Lemma 26 to derive an accurate estimate of $\ln r_{k,w_i}$ for each w_i.

We set $N_t = N - t$ for simplicity. Recall that

$$r_{k,w} = \binom{t}{k}\binom{N_t}{w-k}\Big/\binom{N}{w}.$$

For each w_i, since $|k-t/2| \leq c_1\sqrt{t\ln N}$ and $t = o(N)$, we have

$$\left|w_i - k - \frac{N_t}{2}\right| \leq \left|w_i - \frac{N}{2}\right| + \left|k - \frac{t}{2}\right| \leq u_i\sqrt{N} + c_1\sqrt{t\ln N} = o(N_t^{2/3}),$$

and note that $|w_i - N/2| = |u_i\sqrt{N}| = o(N^{2/3})$. So we can apply Lemma 26 to derive

$$\begin{aligned}
\ln r_{k,w_i} &= \ln\binom{t}{k} + \ln\binom{N_t}{w_i-k} - \ln\binom{N}{w_i} \\
&= -\frac{(w_i-k-N_t/2)^2}{N_t/2} + \frac{(w_i-N/2)^2}{N/2} + C + \ln\binom{t}{k} + o(1),
\end{aligned}$$

in which C is a constant that does not depend on k or w_i.

Let $d = (k - t/2)/\sqrt{t}$ (so $k = t/2 + d\sqrt{t}$), and recall that $w_i = N/2 + u_i\sqrt{N}$ for each w_i. We can further simplify the expression as

$$\begin{aligned}\ln r_{k,w_i} &= -\tfrac{(N/2+u_i\sqrt{N}-t/2-d\sqrt{t}-N_t/2)^2}{N_t/2} + \tfrac{(N/2+u_i\sqrt{N}-N/2)^2}{N/2} + C + \ln\binom{t}{k} + o(1)\\ &= -\tfrac{(u_i\sqrt{N}-d\sqrt{t})^2}{N_t/2} + 2u_i^2 + C + \ln\binom{t}{k} + o(1).\end{aligned}$$

Estimation of $\frac{r_{k,w_j}}{r_{k,w_i}}$. Note that $N_t = N - t = (1 - o(1))N$. So we can approximate the ratio between two r_{k,w_i} and r_{k,w_j} by

$$\begin{aligned}\ln\frac{r_{k,w_j}}{r_{k,w_i}} &= \ln r_{k,w_j} - \ln r_{k,w_i}\\ &= -\frac{(u_j\sqrt{N} - d\sqrt{t})^2}{N_t/2} + 2u_j^2 + \frac{(u_i\sqrt{N} - d\sqrt{t})^2}{N_t/2} - 2u_i^2 + o(1)\\ &= 2u_j^2 - 2u_i^2 + \frac{((u_i+u_j)\sqrt{N} - 2d\sqrt{t})(u_i-u_j)\sqrt{N}}{N_t/2} + o(1)\\ &= 2u_j^2 - 2u_i^2 + 2(u_i^2 - u_j^2) - 4d\frac{\sqrt{tN}}{N_t}(u_i - u_j) + o(1)\\ &= -4d\frac{\sqrt{tN}}{N_t}(u_i - u_j) + o(1).\end{aligned}$$

Verifying (18). Finally, to simplify matters further, we set $x = -4d\frac{\sqrt{tN}}{N_t}$, and substitute it in (19) for k. We have

$$\ln\frac{r_{k,w_0}}{r_{k,w_1}} = x(u_1 - u_0) + o(1) < -\ln 20,$$

which simplifies to

$$x < \frac{-\ln 20}{u_1 - u_0} + o(1) \le \frac{-\ln 20}{0.505} + o(1) \le -5.93 + o(1).$$

Similarly, we have

$$\ln\frac{r_{k,w_2}}{r_{k,w_1}} = x(u_1 - u_2) + o(1) < \ln 10$$

and

$$x > -\frac{\ln 10}{u_2 - u_1} - o(1) \ge -\frac{\ln 10}{0.45} - o(1) \ge -5.12 - o(1).$$

contradiction.

The lower bound. So (18) holds for all valid k, which means for all k such that $|k - t/2| \le c_1\sqrt{t\ln N}$, either $\frac{r_{k,w_0}}{r_{k,w_1}} \ge 0.05$ or $\frac{r_{k,w_2}}{r_{k,w_1}} \ge 10$.

Let H be the set of all valid integers k. We set

$$S = \left\{k \in H : \frac{r_{k,w_0}}{r_{k,w_1}} \ge 0.05\right\} \text{ and } T = H \setminus S.$$

By (18), for any $k \in T$, we have $\frac{r_{k,w_2}}{r_{k,w_1}} \geq 10$.

Since $p_{w_1} = \sum_{k \in S} q_k \cdot r_{k,w_1} + \sum_{k \in T} q_k \cdot r_{k,w_1} \geq \frac{0.89}{N}$ (recall we have set all q_k's to zero for $k \notin H$), we must have either $\sum_{k \in S} q_k \cdot r_{k,w_1} \geq \frac{0.445}{N}$ or $\sum_{k \in T} q_k \cdot r_{k,w_1} \geq \frac{0.445}{N}$.

If $\sum_{k \in S} q_k \cdot r_{k,w_1} \geq \frac{0.445}{N}$, we have

$$p_{w_0} \geq \sum_{k \in S} q_k \cdot r_{k,w_1} \cdot \frac{r_{k,w_0}}{r_{k,w_1}} \geq \frac{0.445}{N} \cdot 0.05 \geq \frac{0.022}{N},$$

which contradicts the constraint that $p_{w_0} \leq \frac{0.02}{N}$. Otherwise, $\sum_{k \in T} q_k \cdot r_{k,w_1} \geq \frac{0.445}{N}$; then

$$p_{w_2} \geq \sum_{k \in T} q_k \cdot r_{k,w_1} \cdot \frac{r_{k,w_2}}{r_{k,w_1}} \geq \frac{0.445}{N} \cdot 10 \geq \frac{4.45}{N},$$

which violates the requirement that $p_{w_2} \leq \frac{4.01}{N}$.

Since both cases lead to a contradiction, A needs to make $\Omega(N)$ queries and this completes the proof. ◀

7 Quantum Supremacy Relative to Efficiently-Computable Oracles

We now discuss our results about quantum supremacy relative to oracles in P/poly.

Building on work by Zhandry [61] and Servedio and Gortler [53], we first show that, if (classical) one-way functions exist, then there exists an oracle $\mathcal{O} \in \mathsf{P/poly}$ such that $\mathsf{BPP}^{\mathcal{O}} \neq \mathsf{BQP}^{\mathcal{O}}$. Then we make a connection to the previous section by showing that, assuming the existence of (classical) subexponentially strong one-way functions, Fourier Fishing and Fourier Sampling are hard even when it is promised that the oracle is in P/poly.

We also study several other complexity questions relative to P/poly oracles: for example, P vs NP, P vs BPP, and BQP vs SZK. Since these questions are not connected directly with quantum supremacy, we will discuss them in Appendix A.

7.1 Preliminaries

Recall that an oracle $\mathcal{O} : \{0,1\}^* \to \{0,1\}$ is itself a language, so we say that an oracle $\mathcal{O}$ is in P/poly when the corresponding language belongs to P/poly, and we use $\mathcal{O}_n$ to denote its restriction to $\{0,1\}^n$.

Given two sets $\mathcal{X}$ and $\mathcal{Y}$, we define $\mathcal{Y}^{\mathcal{X}}$ as the set of functions $f : \mathcal{X} \to \mathcal{Y}$. For a set $\mathcal{X}$, we will sometimes abuse notation and write $\mathcal{X}$ to denote the uniform distribution on $\mathcal{X}$.

(Quantum) Pseudorandom Functions and Permutations

We are going to use pseudorandom functions and permutations throughout this section, so we first review their definitions.

▶ **Definition 34** (PRF and PRP)**.** A pesudorandom function is a function $\mathsf{PRF} : \mathcal{K} \times \mathcal{X} \to \mathcal{Y}$, where $\mathcal{K}$ is the key-space, and $\mathcal{X}$ and $\mathcal{Y}$ are the domain and the range. $\mathcal{K}, domain, image$ are implicitly functions of the security parameter n.[10] We write $y = \mathsf{PRF}_k(x)$.

[10] We denote them by $\mathcal{K}_n, domain_n, image_n$ when we need to be clear about the security parameter n.

Similarly, a pesudorandom permutation is a function $\mathsf{PRP} : \mathcal{K} \times \mathcal{X} \to \mathcal{X}$, where $\mathcal{K}$ is the key-space, and $\mathcal{X}$ is the domain of the permutation. $\mathcal{K}$ and $\mathcal{X}$ are implicitly functions of the security parameter n. We write $y = \mathsf{PRP}_k(x)$. It is guaranteed that PRP_k is a permutation on $\mathcal{X}$ for each $k \in \mathcal{K}$.

For simplicity, we use $\mathsf{PRF}_{\mathcal{K}}$ to denote the distribution on functions $f : \mathcal{X} \to \mathcal{Y}$ by drawing $k \leftarrow \mathcal{K}$ and set $f := \mathsf{PRF}_k$.

We now introduce the definitions of classical and quantum security.

▶ **Definition 35** (Classical-Security)**.** A pseudorandom function $\mathsf{PRF} : \mathcal{K} \times \mathcal{X} \to \mathcal{Y}$ is (classically) secure if no classical adversary A can distinguish between a truly random function and the function PRF_k for a random k in polynomial time. That is, for every such A, there exists a negligible function $\varepsilon = \varepsilon(n)$ such that

$$\left| \Pr_{k \leftarrow \mathcal{K}}[A^{\mathsf{PRF}_k}() = 1] - \Pr_{f \leftarrow \mathcal{Y}^{\mathcal{X}}}[A^{f}() = 1] \right| < \varepsilon.$$

Also, we say that a pseudorandom function PRF is exponentially-secure, if the above holds even for classical adversaries that take $2^{O(n)}$ time.

Similarly, a pseudorandom permutation PRP is (classically) secure if no classical adversary A can distinguish between a truly random permutation and the function PRP_k for a random k in polynomial time.

Sometimes, especially in the context of one-way functions, we will talk about *subexponential* security. By this we simply mean that there is no adversary running in $2^{n^{o(1)}}$ time.

▶ **Definition 36** (Quantum-Security)**.** A pseudorandom function PRF is quantum-secure if no quantum adversary A making quantum queries can distinguish between a truly random function and the function PRF_k for a random k in polynomial time.

Also, a pseudorandom permutation PRP is quantum-secure if no quantum adversary A making quantum queries can distinguish between a truly random permutation and the function PRP_k for a random k in polynomial time.

On the Existence of PRFs

It is well-known that the existence of one-way functions implies the existence of PRFs and PRPs.

▶ **Lemma 37** ([34, 32, 33, 42])**.** *If one-way functions exist, then there exist secure PRFs and PRPs. Similarly, if subexponentially-secure one-way functions exist, then there exist exponentially-secure PRFs.*

We remark here that these are all *purely classical* assumptions, which make no reference to quantum algorithms. Also, the latter assumption is the same one as in the famous *natural proofs* barrier [48].

7.2 A Construction from Zhandry [61]

To prove our separations, we will use a construction from Zhandry [61] with some modifications. We first construct a PRP and a PRF, and summarize some of their useful properties.

Definitions of $\mathsf{PRP}^{\mathsf{raw}}$ and $\mathsf{PRF}^{\mathsf{mod}}$

Assuming one-way functions exist, by Lemma 37, let $\mathsf{PRP}^{\mathsf{raw}}$ be a secure pesudorandom permutation with key-space $\mathcal{K}^{\mathsf{raw}}$ and domain $\mathcal{X}^{\mathsf{raw}}$. We interpret $\mathcal{X}^{\mathsf{raw}}$ as $[N]$, where $N = N(n) = |\mathcal{X}^{\mathsf{raw}}|$.

Then we define another pseudorandom function $\mathsf{PRF}^{\mathsf{mod}}_{(k,a)}(x) = \mathsf{PRP}^{\mathsf{raw}}_k((x-1) \bmod a + 1)$ where:

- The key space of $\mathsf{PRF}^{\mathsf{mod}}$ is $\mathcal{K}^{\mathsf{mod}} = \mathcal{K}^{\mathsf{raw}} \times \mathcal{A}$ where $\mathcal{A}$ is the set of primes in $[\sqrt{N}/4, sqrtN/2]$.
- The domain and image are both $\mathcal{X}^{\mathsf{raw}}$, that is, $\mathcal{X}^{\mathsf{mod}} = \mathcal{X}^{\mathsf{raw}}$ and $\mathcal{Y}^{\mathsf{mod}} = \mathcal{X}^{\mathsf{raw}}$.

Note that we denote the latter one by $\mathsf{PRF}^{\mathsf{mod}}$ (not $\mathsf{PRP}^{\mathsf{mod}}$) because it is no longer a PRP.

Properties of $\mathsf{PRP}^{\mathsf{raw}}$ and $\mathsf{PRF}^{\mathsf{mod}}$

We now summarize several properties of $\mathsf{PRP}^{\mathsf{raw}}$ and $\mathsf{PRF}^{\mathsf{mod}}$, which can be proved along the same lines as [61].

▶ **Lemma 38** (Implicit in Claim 1 and Claim 2 of [61]). *The following statements hold when $\mathsf{PRP}^{\mathsf{raw}}$ is classical secure.*

1. *Both $\mathsf{PRP}^{\mathsf{raw}}$ and $\mathsf{PRF}^{\mathsf{mod}}$ are classical secure PRFs. Consequently, no classical algorithm A can distinguish them with a non-negligible advantage.*
2. *Given oracle access to $\mathsf{PRF}^{\mathsf{mod}}_{(k,a)}$ where $(k,a) \leftarrow \mathcal{K}^{\mathsf{mod}}$, there is a quantum algorithm that can recover a with probability at least $1-\varepsilon$.*
3. *There is a quantum algorithm that can distinguish $\mathsf{PRP}^{\mathsf{raw}}$ from $\mathsf{PRF}^{\mathsf{mod}}$ with advantage $1-\varepsilon$.*

Here $\varepsilon = \varepsilon(n)$ is a negligible function.

For completeness, we prove Lemma 38 in Appendix D, by adapting the proofs of Claims 1 and 2 in [61].

7.3 BPP vs. BQP

Next we discuss whether there is an oracle $\mathcal{O} \in \mathsf{P/poly}$ that separates BPP from BQP. We show that the answer is yes provided that one-way functions exist.

▶ **Theorem 39.** *Assuming one-way functions exist, there exists an oracle $\mathcal{O} \in P/poly$ such that $BPP^{\mathcal{O}} \neq BQP^{\mathcal{O}}$.*

Proof. We are going to use $\mathsf{PRP}^{\mathsf{raw}}$ and $\mathsf{PRF}^{\mathsf{mod}}$ from Section 7.2.

The oracle $\mathcal{O}$ will encode the truth tables of functions $f_1, f_2, dotsc$, where each f_n is a function from $\mathcal{X}^{\mathsf{raw}}_n$ to $\mathcal{X}^{\mathsf{raw}}_n$. For each n, with probability 0.5 we draw f_n from $\mathsf{PRP}^{\mathsf{raw}}_{\mathcal{K}^{\mathsf{raw}}}$, that is, draw $k \leftarrow \mathcal{K}^{\mathsf{raw}}$ and set $f_n := \mathsf{PRP}^{\mathsf{raw}}_k$, and with probability 0.5 we draw f_n from $\mathsf{PRF}^{\mathsf{mod}}_{\mathcal{K}^{\mathsf{mod}}}$ similarly. We set L to be the unary language consisting of all 0^n for which f_n is drawn from $\mathsf{PRP}^{\mathsf{raw}}_{\mathcal{K}^{\mathsf{raw}}}$.

By Lemma 38, there exists a BQP machine $M^{\mathcal{O}}$ that decides L correctly on all but finite many values of n with probability 1. Since we can simply hardwire the values of n on which $M^{\mathcal{O}}$ is incorrect, it follows that $L \in \mathsf{BQP}^{\mathcal{O}}$ with probability 1.

On the other hand, again by Lemma 38, no BPP machine can distinguish $\mathsf{PRP}^{\mathsf{raw}}_{\mathcal{K}^{\mathsf{raw}}}$ and $\mathsf{PRF}^{\mathsf{mod}}_{\mathcal{K}^{\mathsf{mod}}}$ with a non-negligible advantage. So let M be a BPP machine, and let $E_n(M)$ be the event that M decides whether $0^n \in L$ correctly. We have

$$\Pr_{\mathcal{O}}[E_n(M)] = \frac{1}{2} + o(1),$$

even conditioning on events $E_1(M), dotsc, E_{n-1}(M)$. Therefore, we have $\Pr_{\mathcal{O}}[\wedge_{i=1}^{+\infty} E_n(M)] = 0$, which means that a BPP machine M decides L with probability 0. Since there are countably many BPP machines, it follows that $L \notin \mathsf{BPP}^{\mathcal{O}}$ with probability 1. Hence $\mathsf{BPP}^{\mathcal{O}} \neq \mathsf{BQP}^{\mathcal{O}}$ with probability 1.

Finally, note that each f_n has a polynomial-size circuit, and consequently $\mathcal{O} \in \mathsf{P/poly}$. ◀

7.4 Fourier Fishing and Fourier Sampling

Finally, we discuss Fourier Fishing and Fourier Sampling. We are going to show that, assuming the existence of subexponentially-secure one-way functions, Fourier Fishing and Fourier Sampling are hard even when it is promised that the oracle belongs to P/poly.

▶ **Theorem 40.** *Assuming the existence of subexponentially strong one-way functions, there is no polynomial-time classical algorithm that can solve promise-Ffishing with probability*

$$\mathsf{Succ}_R + \Omega(1),$$

even when it is promised that the oracle function belongs to P/poly.

Proof. By Lemma 37, we can use our one-way function to construct an exponentially-secure pseudorandom function, $\mathsf{PRF} : \mathcal{K} \times \mathcal{X} \to \mathcal{Y}$. Without loss of generality, we assume that $|\mathcal{Y}| = 2$ and $|\mathcal{X}| = 2^n$. Then we interpret $\mathcal{X}$ as the set $\{0,1\}^n$, and $\mathcal{Y}$ as the set $\{-1,1\}$.

A Concentration Inequality. Now, consider the distribution $\mathsf{PRF}_{\mathcal{K}}$ on functions $\{0,1\}^n \to \{-1,1\}$. We claim that

$$\Pr_{f \leftarrow \mathsf{PRF}_{\mathcal{K}}}[\mathsf{adv}(f) > \mathsf{Succ}_Q - 1/n] > 1 - \frac{1}{n} - o(1). \tag{20}$$

To see this: from Lemma 30, we have

$$\Pr_{f \leftarrow \mathcal{Y}^{\mathcal{X}}}[\mathsf{adv}(f) > \mathsf{Succ}_Q - 1/n] > 1 - \frac{1}{n}.$$

Therefore, if (20) does not hold, then we can construct a distinguisher between $\mathsf{PRF}_{\mathcal{K}}$ and truly random functions $\mathcal{X}^{\mathcal{Y}}$ by calculating $\mathsf{adv}(f)$ in $2^{O(n)}$ time. But this contradicts the assumption that PRF is exponentially-secure.

A distributional lower bound. Next, we show that for every polynomial-time algorithm A, we have

$$\Pr_{f \leftarrow \mathsf{PRF}_{\mathcal{K}}}[A^f \text{ solves Ffishing correctly}] \leq \mathsf{Succ}_R + o(1). \tag{21}$$

This is because when f is a truly random function, from Lemma 31, we have

$$\Pr_{f \leftarrow \mathcal{Y}^{\mathcal{X}}}[A^f \text{ solve Ffishing correctly}] \leq \mathsf{Succ}_R + o(1).$$

So if (21) does not hold, then we can construct a distinguisher between $\mathsf{PRF}_{\mathcal{K}}$ and truly random functions $\mathcal{X}^{\mathcal{Y}}$ by simulating A^f to get its output z, and then checking whether z is a correct solution to Ffishing in $2^{O(n)}$ time. This again contradicts our assumption that PRF is exponentially-secure.

The lower bound. Finally, we prove the theorem. Suppose for contradiction that there is such a polynomial-time algorithm A. Then when $f \leftarrow \mathsf{PRF}_{\mathcal{K}}$, from (20), with probability $1 - 1/n - o(1)$, we have that f satisfies the promise of promise-Ffishing. Thus, A solves Ffishing when $f \leftarrow \mathsf{PRF}_{\mathcal{K}}$ with probability at least

$$(1 - o(1)) \cdot (\mathsf{Succ}_R + \Omega(1)) = \mathsf{Succ}_R + \Omega(1),$$

which contradicts (21). ◀

By a similar reduction, we can show that Fourier Sampling is also hard.

▶ **Corollary 41.** *Assuming the existence of subexponentially-secure one-way functions, no polynomial-time classical algorithm can solve Fsampling with error*

$$\varepsilon < \mathsf{Succ}_Q - \mathsf{Succ}_R \approx 0.483,$$

even if it is promised that the oracle function belongs to P/poly.

Proof. For a function f, an exact algorithm for Fsampling can be used to solve Ffishing with probability $\mathsf{adv}(f)$. Hence, a polynomial-time sampling algorithm A for Fsampling with error at most ε can solve Ffishing with probability at least $\mathsf{adv}(f) - \varepsilon$.

Note that by (20), when $f \leftarrow \mathsf{PRF}_{\mathcal{K}}$, the algorithm A can solve Ffishing with probability at least

$$(\mathsf{Succ}_Q - \frac{1}{n} - \varepsilon) \cdot (1 - o(1)) = \mathsf{Succ}_Q - o(1) - \varepsilon.$$

Therefore, by (21), we must have $\varepsilon \geq \mathsf{Succ}_Q - \mathsf{Succ}_R$, which completes the proof. ◀

8 Complexity Assumptions Are Needed for Quantum Supremacy Relative to Efficiently-Computable Oracles

In Section 7.4, we showed that the existence of subexponentially-secure one-way functions implies that Fourier Sampling and Fourier Fishing are classically hard, even when it is promised that the oracle function belongs to P/poly. We also showed that if one-way functions exist, then there exists an oracle $\mathcal{O} \in \mathsf{P/poly}$ which separates BPP from BQP.

It is therefore natural to ask whether we can prove the same statements *unconditionally*. In this section, we show that at least *some* complexity assumptions are needed.

▶ **Theorem 42.** *Suppose* $\mathsf{SampBPP} = \mathsf{SampBQP}$ *and* $\mathsf{NP} \subseteq \mathsf{BPP}$. *Then for every oracle* $\mathcal{O} \in \mathsf{P/poly}$, *we have* $\mathsf{SampBPP}^{\mathcal{O}} = \mathsf{SampBQP}^{\mathcal{O}}$ *(and consequently* $\mathsf{BPP}^{\mathcal{O}} = \mathsf{BQP}^{\mathcal{O}}$*).*

Much like in the proof of Theorem 21, we need to show that under the stated assumptions, every SampBQP algorithm M can be simulated by a SampBPP algorithm A.

▶ **Lemma 43.** *Suppose* $\mathsf{SampBPP} = \mathsf{SampBQP}$ *and* $\mathsf{NP} \subseteq \mathsf{BPP}$. *Then for any polynomial* $q(n)$ *and any* SampBQP *oracle algorithm* M, *there is a* SampBPP *oracle algorithm* A *such that:*

For every $\mathcal{O} \in \mathsf{SIZE}(q(n))$,[11] *let* $\mathcal{D}^M_{x,varepsilon}$ *and* $\mathcal{D}^A_{x,varepsilon}$ *be the distributions output by* $M^{\mathcal{O}}$ *and* $A^{\mathcal{O}}$ *respectively on input* $\langle x, 0^{1/\varepsilon} \rangle$. *Then*

$$\|\mathcal{D}^M_{x,varepsilon} - \mathcal{D}^A_{x,varepsilon}\| \leq \varepsilon.$$

[11] A language is in $\mathsf{SIZE}(q(n))$ if it can be computed by circuits of size $q(n)$.

Before proving Lemma 43, we show that it implies Theorem 42.

Proof of Theorem 42. Let $\mathcal{O} \in \mathsf{P/poly}$ be an oracle. Then there exists a polynomial $q(n)$ such that $\mathcal{O} \in \mathsf{SIZE}(q(n))$.

Let $\mathcal{S}$ be a sampling problem in $\mathsf{SampBQP}^{\mathcal{O}}$. This means that there is a $\mathsf{SampBQP}$ oracle algorithm M, such that for all $x \in \{0,1\}^*$ and ε, we have $\|\mathcal{D}^M_{x,varepsilon} - \mathcal{S}_x\| \leq \varepsilon$. Let A_M be the corresponding $\mathsf{SampBPP}$ algorithm whose existence we've assumed, and consider the following algorithm A': given input $\langle x, 0^{1/\varepsilon}\rangle$, run A_M on input $\langle x, 0^{2/\varepsilon}\rangle$ to get a sample from $\mathcal{D}^{A_M}_{x,varepsilon/2}$.

Then we have

$$\begin{aligned}\|\mathcal{D}^{A'}_{x,varepsilon} - \mathcal{S}_x\| &= \|\mathcal{D}^{A_M}_{x,varepsilon/2} - \mathcal{S}_x\| \\ &\leq \|\mathcal{D}^{M}_{x,varepsilon/2} - \mathcal{D}^{A_M}_{x,varepsilon/2}\| + \|\mathcal{D}^{M}_{x,varepsilon/2} - \mathcal{S}_x\| \\ &\leq 2 \cdot \frac{\varepsilon}{2} \leq \varepsilon.\end{aligned}$$

This means that A' solves $\mathcal{S}$ and $\mathcal{S} \in \mathsf{SampBPP}^{\mathcal{O}}$. Hence $\mathsf{SampBQP}^{\mathcal{O}} \subseteq \mathsf{SampBPP}^{\mathcal{O}}$. ◀

Now we prove Lemma 43. The simulation procedure is similar to that in Lemma 23: that is, we replace each oracle gate, one by one, by a known function while minimizing the introduced error. The difference is that, instead of the brute-force method as in Lemma 23, here we use a more sophisticated PAC learning subroutine to find an "approximator" to replace the oracle gates.

Proof of Lemma 43. Let $\mathcal{O} \in \mathsf{SIZE}(q(n))$; we let $f_n = \mathcal{O}_n$ for simplicity.

Recall that there exists a fixed polynomial p, such that given input $\langle x, 0^{1/\varepsilon}\rangle$, the machine M first constructs a quantum circuit C with $N = p(|x|, 1/\varepsilon)$ qubits and N gates classically (C can contain $\mathcal{O}$ gates). Without loss of generality, we can assume for each n, all f_n gates act only on the first n qubits.

For a function $f : \{0,1\}^k \to \{0,1\}$, recall that U_f denotes the unitary operator mapping $|i\rangle$ to $(-1)^{f(i)}|i\rangle$ for $i \in \{0,1\}^k$.

Suppose there are T $\mathcal{O}$-gates in total, and the i-th $\mathcal{O}$-gate is an f_{n_i} gate. Then the unitary operator U applied by the circuit C can be decomposed as

$$U = U_{T+1}(U_{f_{n_T}} \otimes I_{N-n_T}) \cdots (U_{f_{n_2}} \otimes I_{N-n_2}) U_2 (U_{f_{n_1}} \otimes I_{N-n_1}) U_1,$$

where the U_i's are the unitary operators corresponding to the sub-circuits which don't contain an $\mathcal{O}$-gate.

Again, the algorithm proceeds by replacing each $\mathcal{O}$-gate by a much simpler gate one by one, without affecting the resulting quantum state too much, and then simulating the final circuit to get a sample to output.

Replacing the t-th $\mathcal{O}$-gate. Suppose we have already replaced the first $t-1$ $\mathcal{O}$-gates: that is, for each $i \in [t-1]$, we replaced the f_{n_i} gate (the i-th $\mathcal{O}$-gate) with a g_i gate. Now we are going to replace the t-th $\mathcal{O}$-gate.

Let

$$|v\rangle = U_t(U_{g_{t-1}} \otimes I_{N-n_{t-1}}) \cdots (U_{g_2} \otimes I_{N-n_2}) U_2 (U_{g_1} \otimes I_{N-n_1}) U_1 |0\rangle^{\otimes N},$$

which is the quantum state right before the t-th $\mathcal{O}$ gate in the circuit after the replacement.

For brevity, we use f to denote the function f_{n_t}, and we drop the subscript t of n_t when it is clear from context.

Analysis of incurred error. The t-th $\mathcal{O}$-gate is an f gate. If we replace it by a g gate, then the deviation caused to the quantum states is

$$\|U_f \otimes I_{N-n}|v\rangle - U_g \otimes I_{N-n}|v\rangle\| = \|(U_f - U_g) \otimes I_{N-n}|v\rangle\|.$$

Let H be the Hilbert space corresponding to the last $N-n$ qubits, and let $\rho = \mathrm{Tr}_H[|v\rangle\langle v|]$. Then proceeding exactly as in Lemma 23, we have

$$\|((U_f - U_g) \otimes I_{N-n})|v\rangle\|^2 = 4 \cdot \Pr_{i\sim Q}[f(i) \neq g(i)], \tag{22}$$

where Q is the probability on $\{0,1\}^n$ defined by $Q(i) = \langle i|\rho|i\rangle$, and $[f(i) \neq g(i)]$ is the indicator function that takes value 1 when $f(i) \neq g(i)$ and 0 otherwise.

Upper bounding the deviation (22) vis PAC learning. Now, we want to replace f by another function g, so that the deviation term (22) is minimized.

By a standard result of PAC learning (cf. the book of Vapnik [59]), for parameters ε_1 and δ_1, we can take a $\mathrm{poly}(n, varepsilon_1^{-1}, ln\delta_1^{-1})$ number of i.i.d. samples from Q, and then find a function g in $\mathsf{SIZE}(q(n))$ which agrees with f on those samples. Then with probability at least $1-\delta_1$, we will have

$$\Pr_{i\sim Q}[f(i) \neq g(i)] \leq \varepsilon_1.$$

The choice of ε_1 and δ_1 will be made later. In any case, with probability at least $1-\delta_1$, we have

$$\|(U_f - U_g) \otimes I_{N-n}|v\rangle\|^2 \leq 4\varepsilon_1,$$

which in turn implies

$$\|(U_f - U_g) \otimes I_{N-n}|v\rangle\| \leq 2 \cdot \sqrt{\varepsilon_1}.$$

Analysis of the final circuit C^{final}. Suppose that at the end, for each $t \in [T]$, our algorithm has replaced the t-th $\mathcal{O}$-gate with a g_t gate, where g_t is a function from $\{0,1\}^{n_t}$ to $\{0,1\}$. Let C^{final} be the circuit after the replacement. Also, let

$$V = U_{T+1}(U_{g_T} \otimes I_{N-n_T}) \cdots (U_{g_2} \otimes I_{N-n_2})U_2(U_{g_1} \otimes I_{N-n_1})U_1$$

be the unitary operator corresponding to C^{final}.

Now we set $\delta_1 = \dfrac{\varepsilon}{2T}$, and $\varepsilon_1 = \dfrac{\varepsilon^4}{256T^2}$. Then by a union bound over all rounds, and following exactly the same analysis as in Lemma 23, with probability at least $1-T\cdot\delta_1 = 1-\varepsilon/2$, we have

$$\|U|0\rangle^{\otimes N} - V|0\rangle^{\otimes N}\| \leq 2T \cdot \sqrt{\varepsilon_1} = \frac{\varepsilon^2}{8}.$$

Our classical algorithm A then simulates stages 2 and 3 of the $\mathsf{SampBQP}$ algorithm M straightforwardly. It first takes a sample z by measuring $V|0\rangle^{\otimes N}$ in the computational basis, and then outputs $A^{\mathsf{output}}(z)$ as its sample, where A^{output} is the classical algorithm used by M in stage 3.

By Corollary 5, with probability at least $1-\varepsilon/2$, the final distribution $\mathcal{D}$ on which A takes samples satisfies

$$\|\mathcal{D} - \mathcal{D}^M_{x,varepsilon}\| \leq \sqrt{2 \cdot \frac{\varepsilon^2}{8}} = \frac{\varepsilon}{2}.$$

Hence, the outputted distribution $\mathcal{D}^A_{x,varepsilon}$ satisfies

$$\|\mathcal{D}^A_{x,varepsilon} - \mathcal{D}^M_{x,varepsilon}\| \leq \varepsilon.$$

Showing that A is a SampBPP algorithm. We still have to show that A is a SampBPP oracle algorithm. From the previous discussion, A needs to do the following non-trivial computations.

- Taking a polynomial number of samples from Q. This task is in SampBQP (no oracle involved) by definition. By our assumption SampBQP = SampBPP, it can be done in SampBPP.
- Finding a $g \in \mathsf{SIZE}(q(n))$ such that g agrees with f on all the samples. This can be done in NP, so by our assumption NP $\subseteq$ BPP, it can be done in BPP.
- Taking a sample by measuring $V|0\rangle^{\otimes N}$. Again, this task is in SampBQP, and hence can be done in SampBPP by our assumption.

Therefore, A is a SampBPP oracle algorithm. ◀

9 Open Problems

There are many exciting open problems left by this paper; here we mention just a few.

1. Is QUATH (our assumption about the hardness of guessing whether $|\langle 0|C|0\rangle|^2$ is greater or less than the median) true or false?
2. Is Conjecture 1 true? That is, does a random quantum circuit on n qubits sample an unbalanced distribution over n-bit strings with $1 - 1/\exp(n)$ probability?
3. We showed that there exists an oracle relative to which SampBPP = SampBQP but PH is infinite. Can we nevertheless show that SampBPP = SampBQP would collapse PH in the unrelativized world? (An affirmative answer would, of course, follow from Aaronson and Arkhipov's Permanent-of-Gaussians Conjecture [3], as mentioned in Section 1.2.)
4. Is our classical algorithm to simulate a quantum circuit with n qubits and m gates optimal? Or could we reduce the complexity, say from $m^{O(n)}$ to $2^{O(n)} \cdot m^{O(1)}$, while keeping the space usage polynomial? Does it matter if we only want to sample from the output distribution, rather than actually calculating the probabilities? What about if we only want to guess an amplitude with small bias, as would be needed to refute QUATH?
5. For random quantum circuit sampling, we proved a conditional hardness result that talks directly about the observed outputs of a sampling process, rather than about the unknown distribution that's sampled from. Can we get analogous hardness results for the BosonSampling or IQP models, under some plausible hardness conjecture? Note that the argument from Section 3 doesn't work directly for BosonSampling or IQP, for the simple reason that in those models, the advantage over chance in guessing a given amplitude is at least $1/\exp(n)$, rather than $1/\exp(m)$ for some $m \gg n$ as is the case for random circuits.
6. We proved a lower bound of $\Omega(N)$ on the classical query complexity of Fourier Sampling, for a rather small error $\varepsilon = \frac{1}{40000}$. The error constant does matter for sampling problems, since there is no efficient way to reduce the error in general. So can we discover the *exact threshold* ε for an $\Omega(N)$ lower bound? That is, find the constant ε such that there is an $o(N)$ query classical algorithm solving Fourier Sampling with error ε, but any classical algorithm with error $< \varepsilon$ needs $\Omega(N)$ queries?
7. In Section 7, we showed that there is an oracle $\mathcal{O}$ in P/poly separating BPP from BQP, assuming that one-way functions exist. Is it possible to weaken the assumption to, say, NP $\not\subset$ BPP?

Acknowledgments We thank Shalev Ben-David, Sergio Boixo, Yuzhou Gu, Greg Kuperberg, John Martinis, Ashley Montanaro, John Preskill, Vadim Smelyansky, Ronald de Wolf, and Mark Zhandry for helpful discussions about the subject of this paper.

References

1 S. Aaronson. BQP and the polynomial hierarchy. In *Proc. ACM STOC*, 2010. arXiv:0910.4698.
2 S. Aaronson. Google, D-wave, and the case of the factor-10^8 speedup for WHAT?, 2015. URL: `http://www.scottaaronson.com/blog/?p=2555`.
3 S. Aaronson and A. Arkhipov. The computational complexity of linear optics. *Theory of Computing*, 9(4):143–252, 2013. Earlier version in Proc. ACM STOC'2011. ECCC TR10-170, arXiv:1011.3245.
4 S. Aaronson et al. The Complexity Zoo. URL: `http://www.complexityzoo.com`.
5 S. Aaronson and Y. Shi. Quantum lower bounds for the collision and the element distinctness problems. *J. of the ACM*, 51(4):595–605, 2004.
6 S. Aaronson and A. Wigderson. Algebrization: a new barrier in complexity theory. *ACM Trans. on Computation Theory*, 1(1), 2009. Earlier version in Proc. ACM STOC'2008.
7 Sccot Aaronson. New evidence that quantum mechanics is hard to simulate on classical computers. `http://www.scottaaronson.com/talks/newev.ppt`, 2009.
8 Scott Aaronson. The equivalence of sampling and searching. *Theory of Computing Systems*, 55(2):281–298, 2014.
9 Scott Aaronson and Andris Ambainis. Forrelation: A problem that optimally separates quantum from classical computing. In *Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing*, pages 307–316. ACM, 2015.
10 Scott Aaronson, Shalev Ben-David, and Robin Kothari. Separations in query complexity using cheat sheets. *arXiv preprint arXiv:1511.01937*, 2015.
11 Scott Aaronson, Adam Bouland, Greg Kuperberg, and Saeed Mehraban. The computational complexity of ball permutations. *arXiv preprint arXiv:1610.06646*, 2016.
12 D. Aharonov and M. Ben-Or. Fault-tolerant quantum computation with constant error. In *Proc. ACM STOC*, pages 176–188, 1997. quant-ph/9906129.
13 D. Aharonov, M. Ben-Or, and E. Eban. Interactive proofs for quantum computations. arXiv:0810.5375, 2008.
14 Andris Ambainis. Polynomial degree and lower bounds in quantum complexity: Collision and element distinctness with small range. *Theory of Computing*, 1(1):37–46, 2005.
15 Sanjeev Arora and Boaz Barak. *Computational complexity: a modern approach*. Cambridge University Press, 2009.
16 Theodore Baker, John Gill, and Robert Solovay. Relativizations of the P=?NP question. *SIAM Journal on computing*, 4(4):431–442, 1975.
17 C. Bennett, E. Bernstein, G. Brassard, and U. Vazirani. Strengths and weaknesses of quantum computing. *SIAM J. Comput.*, 26(5):1510–1523, 1997. quant-ph/9701001.
18 E. Bernstein and U. Vazirani. Quantum complexity theory. *SIAM J. Comput.*, 26(5):1411–1473, 1997. Earlier version in Proc. ACM STOC'1993.
19 Sergio Boixo, Sergei V Isakov, Vadim N Smelyanskiy, Ryan Babbush, Nan Ding, Zhang Jiang, John M Martinis, and Hartmut Neven. Characterizing quantum supremacy in near-term devices. *arXiv preprint arXiv:1608.00263*, 2016.
20 Dan Boneh and Richard J. Lipton. Quantum cryptanalysis of hidden linear functions. In *Annual International Cryptology Conference*, pages 424–437. Springer, 1995.
21 Fernando G. S. L. Brandão, Aram W. Harrow, and Michał Horodecki. Local random quantum circuits are approximate polynomial-designs. *Communications in Mathematical Physics*, 346(2):397–434, 2016.

22 Sergey Bravyi and David Gosset. Improved classical simulation of quantum circuits dominated by clifford gates. *arXiv preprint arXiv:1601.07601*, 2016.

23 M. Bremner, R. Jozsa, and D. Shepherd. Classical simulation of commuting quantum computations implies collapse of the polynomial hierarchy. *Proc. Roy. Soc. London*, A467(2126):459–472, 2010. arXiv:1005.1407.

24 Michael J. Bremner, Ashley Montanaro, and Dan J. Shepherd. Average-case complexity versus approximate simulation of commuting quantum computations. *arXiv preprint arXiv:1504.07999*, 2015.

25 Michael J. Bremner, Ashley Montanaro, and Dan J. Shepherd. Achieving quantum supremacy with sparse and noisy commuting quantum computations. *arXiv preprint arXiv:1610.01808*, 2016.

26 A. Broadbent, J. Fitzsimons, and E. Kashefi. Universal blind quantum computation. In *Proc. IEEE FOCS*, 2009. arXiv:0807.4154.

27 Jacques Carolan, Christopher Harrold, Chris Sparrow, Enrique Martín-López, Nicholas J Russell, Joshua W Silverstone, Peter J Shadbolt, Nobuyuki Matsuda, Manabu Oguma, Mikitaka Itoh, Graham D Marshall, Mark G Thompson, Jonathan C F Matthews, Toshikazu Hashimoto, Jeremy L O'Brien, and Anthony Laing. Universal linear optics. *Science*, 349(6249):711–716, 2015.

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

29 Edward Farhi and Aram W. Harrow. Quantum supremacy through the quantum approximate optimization algorithm. *arXiv preprint arXiv:1602.07674*, 2016.

30 L. Fortnow and J. Rogers. Complexity limitations on quantum computation. *J. Comput. Sys. Sci.*, 59(2):240–252, 1999. cs.CC/9811023.

31 Keisuke Fujii. Noise threshold of quantum supremacy. *arXiv preprint arXiv:1610.03632*, 2016.

32 O. Goldreich, S. Goldwasser, and S. Micali. How to construct random functions. *J. of the ACM*, 33(4):792–807, 1986. Earlier version in Proc. IEEE FOCS'1984, pp. 464-479.

33 Oded Goldreich and Leonid A Levin. A hard-core predicate for all one-way functions. In *Proceedings of the twenty-first annual ACM symposium on Theory of computing*, pages 25–32. ACM, 1989.

34 J. Håstad, R. Impagliazzo, L. A. Levin, and M. Luby. A pseudorandom generator from any one-way function. *SIAM J. Comput.*, 28(4):1364–1396, 1999.

35 Johan Hastad. Almost optimal lower bounds for small depth circuits. In *Proceedings of the eighteenth annual ACM symposium on Theory of computing*, pages 6–20. ACM, 1986.

36 R. Impagliazzo and A. Wigderson. P=BPP unless E has subexponential circuits: derandomizing the XOR Lemma. In *Proc. ACM STOC*, pages 220–229, 1997.

37 Richard Jozsa and Marrten Van den Nest. Classical simulation complexity of extended clifford circuits. *Quantum Information & Computation*, 14(7&8):633–648, 2014.

38 Gil Kalai. How quantum computers fail: quantum codes, correlations in physical systems, and noise accumulation. *arXiv preprint arXiv:1106.0485*, 2011.

39 J. Kelly, R. Barends, A. G. Fowler, A. Megrant, E. Jeffrey, T. C. White, D. Sank, J. Y. Mutus, B. Campbell, Yu Chen, et al. State preservation by repetitive error detection in a superconducting quantum circuit. *Nature*, 519(7541):66–69, 2015.

40 Samuel Kutin. Quantum lower bound for the collision problem with small range. *Theory of Computing*, 1(1):29–36, 2005.

41 Leonid A. Levin. The tale of one-way functions. *Problems of Information Transmission*, 39(1):92–103, 2003.

42 Michael Luby and Charles Rackoff. How to construct pseudorandom permutations from pseudorandom functions. *SIAM Journal on Computing*, 17(2):373–386, 1988.

43 Igor L. Markov and Yaoyun Shi. Simulating quantum computation by contracting tensor networks. *SIAM Journal on Computing*, 38(3):963–981, 2008.

44 Tomoyuki Morimae, Keisuke Fujii, and Joseph F Fitzsimons. Hardness of classically simulating the one-clean-qubit model. *Physical review letters*, 112(13):130502, 2014.

45 Noam Nisan and Avi Wigderson. Hardness vs randomness. *Journal of computer and System Sciences*, 49(2):149–167, 1994.

46 Borja Peropadre, Gian Giacomo Guerreschi, Joonsuk Huh, and Alán Aspuru-Guzik. Microwave boson sampling. *arXiv preprint arXiv:1510.08064*, 2015.

47 John Preskill. Quantum computing and the entanglement frontier. *arXiv preprint arXiv:1203.5813*, 2012.

48 A. A. Razborov and S. Rudich. Natural proofs. *J. Comput. Sys. Sci.*, 55(1):24–35, 1997. Earlier version in Proc. ACM STOC'1994, pp. 204-213.

49 Benjamin Rossman, Rocco A Servedio, and Li-Yang Tan. An average-case depth hierarchy theorem for boolean circuits. In *Foundations of Computer Science (FOCS), 2015 IEEE 56th Annual Symposium on*, pages 1030–1048. IEEE, 2015.

50 Terry Rudolph. Why I am optimistic about the silicon-photonic route to quantum computing. *arXiv preprint arXiv:1607.08535*, 2016.

51 Amit Sahai and Salil Vadhan. A complete problem for statistical zero knowledge. *Journal of the ACM (JACM)*, 50(2):196–249, 2003.

52 W. J. Savitch. Relationships between nondeterministic and deterministic tape complexities. *J. Comput. Sys. Sci.*, 4(2):177–192, 1970.

53 Rocco A Servedio and Steven J Gortler. Equivalences and separations between quantum and classical learnability. *SIAM Journal on Computing*, 33(5):1067–1092, 2004.

54 A. Shamir. IP=PSPACE. *J. of the ACM*, 39(4):869–877, 1992. Earlier version in Proc. IEEE FOCS'1990, pp. 11-15.

55 P. W. Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM J. Comput.*, 26(5):1484–1509, 1997. Earlier version in Proc. IEEE FOCS'1994. quant-ph/9508027.

56 Joel Spencer. *Asymptopia*, volume 71. American Mathematical Soc., 2014.

57 B. M. Terhal and D. P. DiVincenzo. Adaptive quantum computation, constant-depth circuits and Arthur-Merlin games. *Quantum Information and Computation*, 4(2):134–145, 2004. quant-ph/0205133.

58 S. Toda. PP is as hard as the polynomial-time hierarchy. *SIAM J. Comput.*, 20(5):865–877, 1991. Earlier version in Proc. IEEE FOCS'1989, pp. 514-519.

59 Vladimir Naumovich Vapnik. *Statistical learning theory*, volume 1. Wiley New York, 1998.

60 Andrew Chi-Chih Yao. Separating the polynomial-time hierarchy by oracles. In *26th Annual Symposium on Foundations of Computer Science (sfcs 1985)*, 1985.

61 Mark Zhandry. How to construct quantum random functions. In *Foundations of Computer Science (FOCS), 2012 IEEE 53rd Annual Symposium on*, pages 679–687. IEEE, 2012.

62 Mark Zhandry. A note on quantum-secure prps. *arXiv preprint arXiv:1611.05564*, 2016.

A Other Results on Oracle Separations in P/poly

In this section we discuss the rest of our results on complexity theory relative to oracles in $\mathsf{P/poly}$ (see Figure 1 for an overview). For the definitions of the involved complexity classes, see for example [4].

We first discuss P and NP. We observe that there exists an oracle $\mathcal{O} \in \mathsf{P/poly}$ such that $\mathsf{P}^{\mathcal{O}} \neq \mathsf{NP}^{\mathcal{O}}$ *unconditionally*, and no oracle $\mathcal{O} \in \mathsf{P/poly}$ can make $\mathsf{P} = \mathsf{NP}$ unless $\mathsf{NP} \subset \mathsf{P/poly}$.

Then we discuss P and BPP. We first prove that the standard derandomization assumption (there exists a function $f \in \mathsf{E} = \mathsf{DTIME}(2^{O(n)})$ that requires a $2^{\Omega(n)}$-size circuit) also implies

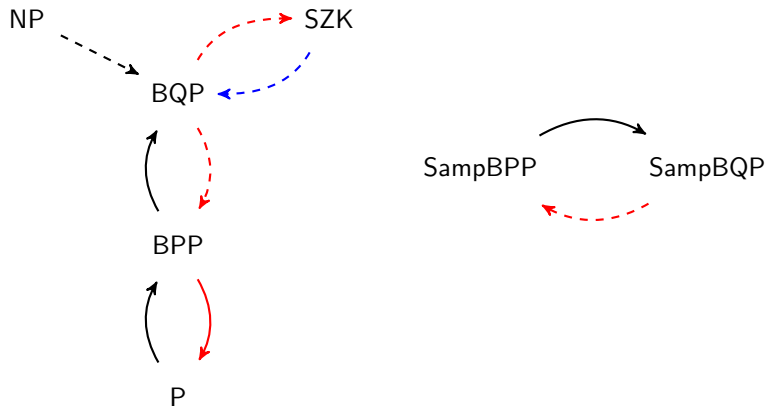

Figure 1 $\mathcal{C}_1 \to \mathcal{C}_2$ indicates $\mathcal{C}_1$ is contained in $\mathcal{C}_2$ respect to every oracle in P/poly, and $\mathcal{C}_1 \dashrightarrow \mathcal{C}_2$ denotes that there is an oracle $\mathcal{O} \in \mathsf{P/poly}$ such that $\mathcal{C}_1^{\mathcal{O}} \not\subset \mathcal{C}_2^{\mathcal{O}}$. Red indicates this statement is based on the existence of *classical one-way functions*, Blue indicates the statement is based on the existence of *quantum one-way functions*, and Black indicates the statement holds unconditionally.

that $\mathsf{P}^{\mathcal{O}} = \mathsf{BPP}^{\mathcal{O}}$ for all $\mathcal{O} \in \mathsf{P/poly}$. Then, surprisingly, we show that the converse also holds! I.e., if no such f exists, then there exists an oracle $\mathcal{O} \in \mathsf{P/poly}$ such that $\mathsf{P}^{\mathcal{O}} \neq \mathsf{BPP}^{\mathcal{O}}$.

Finally, we discuss BQP and SZK. We show that assuming the existence of one-way functions, there exist oracles in P/poly that separate BQP from SZK, and also SZK from BQP.

We will need to use quantum-secure pseudorandom permutations. By a very recent result of Zhandry [62], their existence follows from the existence of quantum one-way functions.

▶ **Lemma 44** ([62]). *Assuming quantum one way functions exist, there exist quantum-secure PRPs.*

A.1 P, BPP, BQP vs. NP

We begin with the relationships of P, BPP, and BQP to NP relative to oracles in P/poly.

The first observation is that using the function OR and standard diagonalization techniques, together with the fact that OR is hard for quantum algorithms [17], we immediately have:

▶ **Observation 45.** *There is an oracle $\mathcal{O} \in P/poly$ such that $NP^{\mathcal{O}} \not\subset BQP^{\mathcal{O}}$.*

On the other side, we also show that unless $\mathsf{NP} \subset \mathsf{P/poly}$ ($\mathsf{BQP/poly}$), there is no oracle $\mathcal{O} \in \mathsf{P/poly}$ such that $\mathsf{NP}^{\mathcal{O}} \subseteq \mathsf{BPP}^{\mathcal{O}}$ ($\mathsf{BQP}^{\mathcal{O}}$).

▶ **Theorem 46.** *Unless $NP \subset P/poly$, there is no oracle $\mathcal{O} \in P/poly$ such that $NP^{\mathcal{O}} \subseteq BPP^{\mathcal{O}}$. Likewise, there is no oracle $\mathcal{O} \in P/poly$ such that $NP^{\mathcal{O}} \subseteq BQP^{\mathcal{O}}$ unless $NP \subseteq BQP/poly$.*

Proof. Suppose there is an oracle $\mathcal{O} \in \mathsf{P/poly}$ such that $\mathsf{NP}^{\mathcal{O}} \subseteq \mathsf{BPP}^{\mathcal{O}}$. Since $\mathsf{BPP} \subset \mathsf{P/poly}$, and $\mathsf{P}^{\mathcal{O}}/\mathsf{poly} \subseteq \mathsf{P/poly}$ (since the relevant parts of the oracle $\mathcal{O}$ can be directly supplied to the P/poly algorithm), we have $\mathsf{NP} \subseteq \mathsf{NP}^{\mathcal{O}} \subset \mathsf{P/poly}$. The second claim can be proved in the same way. ◀

The following corollary is immediate.

▶ **Corollary 47.** *There is an oracle $\mathcal{O} \in P/poly$ such that $P^{\mathcal{O}} \neq NP^{\mathcal{O}}$, and there is no oracle $\mathcal{O} \in P/poly$ such that $P^{\mathcal{O}} = NP^{\mathcal{O}}$ unless $NP \subset P/poly$.*

A.2 P vs. BPP

Next we consider the relationship between P and BPP. It is not hard to observe that the standard derandomization assumption for P = BPP is in fact strong enough to make $\mathsf{P}^{\mathcal{O}} = \mathsf{BPP}^{\mathcal{O}}$ for every oracle $\mathcal{O}$ in P/poly.

Given a function $f : \{0,1\}^n \to \{0,1\}$, let $H_{\text{wrs}}(f)$ be the minimum size of circuits computing f exactly.

▶ **Observation 48** (Implicit in [45, 36], see also Theorem 20.7 in [15])**.** *If there exists a function $f \in E = DTIME(2^{O(n)})$ and $\varepsilon > 0$ such that $H_{wrs}(f) \geq 2^{\varepsilon n}$ for sufficiently large n, then $BPP^{\mathcal{O}} = P^{\mathcal{O}}$ for every $\mathcal{O} \in P/poly$.*

Proof Sketch. From [45] and [36], the assumption leads to a strong PRG which is able to fool circuits of a fixed polynomial size with a logarithmic seed length.

An algorithm with an oracle $\mathcal{O} \in \mathsf{P/poly}$ with a certain input can still be represented by a polynomial size circuit, so we can still enumerate all possible seeds to get a deterministic algorithm. ◀

Surprisingly, we show that condition is not only sufficient, but also necessary.

▶ **Theorem 49.** *If for every $f \in E = DTIME(2^{O(n)})$ and $\varepsilon > 0$, there are infinitely many n's with $H_{wrs}(f) < 2^{\varepsilon n}$, then there exists an oracle $\mathcal{O} \in P/poly$ such that $BPP^{\mathcal{O}} \neq P^{\mathcal{O}}$.*

Proof. For simplicity, in the following we will specify an oracle $\mathcal{O}$ by a sequence of functions $\{f_i\}$, where each f_i is a function from $\{0,1\}^{n_i} \to \{0,1\}$ and the sequence $\{n_i\}$ is strictly increasing. That is, $\mathcal{O}_{n_i}$ is set to f_i, and $\mathcal{O}$ maps all strings with length not in $\{n_i\}$ to 0.

As there are only countably many P oracle TM machines, we let $\{A_i\}_{i=1}^{+\infty}$ be an ordering of them.

The GapMaj function. Recall that the gapped-majority function, $\mathsf{GapMaj} : \{0,1\}^N \to \{0,1\}$, which outputs 1 if the input has Hamming weight $\geq 2N/3$, or 0 if the input has Hamming weight $\leq N/3$, and is undefined otherwise, is the function which separates P and BPP in the query complexity world. We are going to encode inputs to GapMaj in the oracle bits to achieve our separation.

We call an oracle *valid*, if for each n, either $|\mathcal{O}_n^{-1}(0)| \geq \frac{2}{3} \cdot 2^n$ or $|\mathcal{O}_n^{-1}(1)| \geq \frac{2}{3} \cdot 2^n$. That is, if we interpret $\mathcal{O}_n$ as a binary string with length 2^n, then $\mathsf{GapMaj}(\mathcal{O}_n)$ is defined.

The language $L^{\mathcal{O}}$. For a valid oracle $\mathcal{O}$, we define the following language:

$$L_{\mathcal{O}} = \{0^n : \mathsf{GapMaj}(\mathcal{O}_n) = 1\}.$$

Clearly, this language lies in $\mathsf{BPP}^{\mathcal{O}}$. To prove the theorem, we will construct a valid oracle $\mathcal{O}$ such that $L_{\mathcal{O}} \notin \mathsf{P}^{\mathcal{O}}$.

Construction of $\mathcal{O}$. To construct such an oracle, we resort to the standard diagonalization method: for each integer i, we find an integer n_i and set the function $\mathcal{O}_{n_i}$ so that the machine A_i can't decide 0^{n_i} correctly. In order to do this, we will make sure that each A_i can only see 0 when querying the function $\mathcal{O}_{n_i}$. Since A_i can only see a polynomial number of bits, we can set the remaining bits in $\mathcal{O}_{n_i}$ adversarially.

Let $\mathcal{O}^i_{\mathsf{part}}$ be the oracle specified by $\{\mathcal{O}_{n_j}\}_{j=1}^i$, and let T_i be the maximum integer such that a bit in $\mathcal{O}_{T_i}$ is queried by A_i when running on input 0^{n_i}. Observe that by setting $n_{i+1} > T_i$, we can make sure that $A_i^{\mathcal{O}}(0^{n_i}) = A_i^{\mathcal{O}^i_{\mathsf{part}}}(0^{n_i})$ for each i.

Diagonalization against A_i. Suppose we have already constructed $\mathcal{O}_{n_1}, dotsc, oracle_{n_{i-1}}$, and we are going to deal with A_i. Since A_i is a P machine, there exists a constant c such that A_i runs in at most n^c steps for inputs with length n. Thus, A_i can query at most n^c values in $\mathcal{O}_n$ on input 0^n.

Construction and Analysis of f. Now consider the following function f, which analyzes the behavior of $A_i^{\mathcal{O}_{\mathsf{part}}^{i-1}}$:

- given an input $x \in \{0,1\}^*$, let $m = |x|$;
- the first $m_1 = \lfloor m/5c \rfloor$ bits of x encode an integer $n \in [2^{m_1}]$;
- the next $m_2 = m - m_1$ bits of x encode a string $p \in \{0,1\}^{m_2}$;
- $f(x) = 1$ iff $A_i^{\mathcal{O}_{\mathsf{part}}^{i-1}}(0^n)$ has queried $\mathcal{O}_n(z)$ for an $z \in \{0,1\}^n$ with p as a prefix.[12]

It is not hard to see that $f \in \mathsf{E}$: the straightforward algorithm which directly simulates $A_i^{\mathcal{O}_{\mathsf{part}}^{i-1}}(0^n)$ runs in $O(n^c) = 2^{O(m/5c \cdot c)} = 2^{O(m)}$ time (note that the input length is $m = |x|$). Therefore, by our assumption, there exists an integer m such that $2^{\lfloor m/5c \rfloor} > \max(T_{i-1}, n_{i-1})$ and $H_{\mathrm{wrs}}(f_m) < 2^{m/c}$. Then we set $n_i = 2^{\lfloor m/5c \rfloor}$.

Construction and Analysis of $\mathcal{O}_{n_i}$. Now, if $A_i^{\mathcal{O}_{\mathsf{part}}^{i-1}}(0^{n_i}) = 1$, we set $\mathcal{O}_{n_i}$ to be the constant function $\mathbf{0}$, so that $L^{\mathcal{O}}(0^{n_i}) = 0$.

Otherwise, $A_i^{\mathcal{O}_{\mathsf{part}}^{i-1}}(0^{n_i}) = 0$. We define a function $g : \{0,1\}^{n_i} \to \{0,1\}$ as follows: $g(z) = 1$ iff $A_i^{\mathcal{O}_{\mathsf{part}}^{i-1}}(0^{n_i})$ has queried $\mathcal{O}_{n_i}(z')$ for an $z' \in \{0,1\}^{n_i}$ such that z and z' share a prefix of length $m - \lfloor m/5c \rfloor$. Note that $g(z)$ can be implemented by hardwiring n_i and $z_{1\ldots m-\lfloor m/5c \rfloor}$ (that is, the first $m - \lfloor m/5c \rfloor$ bits of z) into the circuit for f_m, which means that there is a circuit of size $2^{m/c} = n_i^{O(1)}$ for g. We set $\mathcal{O}_{n_i} := \neg g$.

From the definition of g and the fact that $A_i^{\mathcal{O}_{\mathsf{part}}^{i-1}}(0^{n_i})$ makes at most n_i^c queries, there is at most a

$$\frac{n_i^c}{2^{m-\lfloor m/5c \rfloor}} < \frac{n_i^c}{2^{4c\lfloor m/5c \rfloor}} = n_i^{-3c}$$

fraction of inputs that are 0 in $\neg g$. Hence, $\mathsf{GapMaj}(\neg g) = 1$ and $L^{\mathcal{O}}(0^{n_i}) = 1$.

We claim that in both cases, we have $A_i^{\mathcal{O}_{\mathsf{part}}^{i-1}}(0^{n_i}) = A_i^{\mathcal{O}_{\mathsf{part}}^{i}}(0^{n_i})$. This holds trivially in the first case since we set $\mathcal{O}_{n_i} := \mathbf{0}$. For the second case, note from the definition of g that all queries by $A_i^{\mathcal{O}_{\mathsf{part}}^{i}}(0^{n_i})$ to $\mathcal{O}_{n_i}$ return 0, and hence A_i will behave exactly the same.

Finally, since we set $n_i > T_{i-1}$ for each i, we have $A_i^{\mathcal{O}}(0^{n_i}) = A_i^{\mathcal{O}_{\mathsf{part}}^{i}}(0^{n_i}) \neq L^{\mathcal{O}}(0^{n_i})$, which means that no A_i can decide $L^{\mathcal{O}}$. ◀

A.3 BQP vs. SZK

Next we investigate the relationship between BQP and SZK relative to oracles in $\mathsf{P/poly}$. We first show that, by using quantumly-secure pseudorandom permutations, as well as the quantum lower bound for distinguishing permutations from 2-to-1 functions [5], we can construct an oracle in $\mathsf{P/poly}$ which separates SZK from BQP.

▶ **Theorem 50.** *Assuming quantum-secure one way functions exist, there exists an oracle $\mathcal{O} \in P/poly$ such that $SZK^{\mathcal{O}} \not\subset BQP^{\mathcal{O}}$.*

[12] For simplicity, we still use $\mathcal{O}_n$ to denote the restriction of $\mathcal{O}_{\mathsf{part}}^{i-1}$ on $\{0,1\}^n$.

Proof. Let PRP be a quantum-secure pseudorandom permutation from $\mathcal{K} \times \mathcal{X} \to \mathcal{X}$, whose existence is guaranteed by Lemma 44.

We first build a pseudorandom 2-to-1 function from PRP. We interpret $\mathcal{X}$ as $[N]$ where $N = |\mathcal{X}|$, and assume that N is even. We construct $\mathsf{PRF}^{2\to1} : (\mathcal{K} \times \mathcal{K}) \times \mathcal{X} \to \mathcal{X}$ as follows:

- The key space $\mathcal{K}^{2\to1}$ is $\mathcal{K} \times \mathcal{K}$. That is, a key $k \in \mathcal{K}^{2\to1}$ is a pair of keys (k_1, k_2).
- $\mathsf{PRF}^{2\to1}_{(k_1,k_2)}(x) := \mathsf{PRP}_{k_2}((\mathsf{PRP}_{k_1}(x) \bmod N/2) + 1)$.

Note that $\mathsf{PRF}^{2\to1}$ would be a uniformly random 2-to-1 function from $[N] \to [N]$, if PRP_{k_1} and PRP_{k_2} were replaced by two uniformly random permutations on $[N]$. Hence, by a standard reduction argument, $\mathsf{PRF}^{2\to1}$ is a quantumly-secure pseudorandom 2-to-1 function. That is, for any polynomial-time quantum algorithm A, we have

$$\left| \Pr_{k\leftarrow\mathcal{K}^{2\to1}}[A^{\mathsf{PRF}^{2\to1}_k}() = 1] - \Pr_{f\leftarrow\mathsf{F}^{2\to1}_{\mathcal{X}}}[A^f() = 1] \right| < \varepsilon,$$

where ε is a negligible function and $\mathsf{F}^{2\to1}_{\mathcal{X}}$ is the set of 2-to-1 functions from $\mathcal{X} \to \mathcal{X}$.

Also, from the definition of PRP, we have

$$\left| \Pr_{k\leftarrow\mathcal{K}^{\mathsf{PRP}}}[A^{\mathsf{PRP}_k}() = 1] - \Pr_{f\leftarrow\mathsf{Perm}_{\mathcal{X}}}[A^f() = 1] \right| < \varepsilon,$$

where $\mathsf{Perm}_{\mathcal{X}}$ is the set of permutations on $\mathcal{X}$.

From the results of Aaronson and Shi [5], Ambainis [14] and Kutin [40], no $o(N^{1/3})$-query quantum algorithm can distinguish a random permutation from a random 2-to-1 function. Therefore, we have

$$\left| \Pr_{f\leftarrow\mathsf{F}^{2\to1}_{\mathcal{X}}}[A^f() = 1] - \Pr_{f\leftarrow\mathsf{Perm}_{\mathcal{X}}}[A^f() = 1] \right| < o(1).$$

Putting the above three inequalities together, we have

$$\left| \Pr_{k\leftarrow\mathcal{K}^{2\to1}}[A^{\mathsf{PRF}^{2\to1}_k}() = 1] - \Pr_{k\leftarrow\mathcal{K}^{\mathsf{PRP}}}[A^{\mathsf{PRP}_k}() = 1] \right| < o(1),$$

which means A cannot distinguish $\mathsf{PRF}^{2\to1}_{\mathcal{K}^{2\to1}}$ and $\mathsf{PRP}_{\mathcal{K}^{\mathsf{PRP}}}$.

On the other side, an SZK algorithm can easily distinguish a permutation from a two-to-one function. Therefore, we can proceed exactly as in Theorem 39 to construct an oracle $\mathcal{O} \in \mathsf{P/poly}$ such that $\mathsf{SZK}^{\mathcal{O}} \not\subset \mathsf{BQP}^{\mathcal{O}}$. ◀

Very recently, Chen [28] showed that, based on a construction similar to the "cheat-sheet" function by Aaronson, Ben-David and Kothari [10], we can take any function which is hard for BPP algorithms, and turn it into a function which is hard for SZK algorithms in a black-box fashion. We are going to adapt this construction, together with a PRF, to build an oracle in $\mathsf{P/poly}$ which separates BQP from SZK.

▶ **Theorem 51.** *Assuming one-way functions exist, there exists an oracle $\mathcal{O} \in P/poly$ such that $BQP^{\mathcal{O}} \not\subset SZK^{\mathcal{O}}$.*

Proof. We will use the $\mathsf{PRF}^{\mathsf{mod}} : \mathcal{K}^{\mathsf{mod}} \times \mathcal{X}^{\mathsf{mod}} \to \mathcal{X}^{\mathsf{mod}}$ defined in Section 7.2 here. For simplicity, we will use $\mathcal{X}$ to denote $\mathcal{X}^{\mathsf{mod}}$ in this proof. Recall that $\mathcal{X}$ is interpreted as $[N]$ for $N = N(n) = |\mathcal{X}|$.

Construction of distributions $\mathcal{D}_n^i$. For each n, we define distributions $\mathcal{D}_n^0$ and $\mathcal{D}_n^1$ on $(\mathcal{X}_n \to \mathcal{X}_n) \times \{0,1\}^{\sqrt{N}/2}$ as follows. We draw a function $f_n : \mathcal{X} \to \mathcal{X}$ from $\mathsf{PRF}^{\mathsf{mod}}_{\mathcal{K}^{\mathsf{mod}}}$, that is, we draw $(k,a) \leftarrow \mathcal{K}^{\mathsf{mod}} = \mathcal{K}^{\mathsf{raw}} \times A$, and set $f_n := \mathsf{PRF}^{\mathsf{mod}}_{(k,a)}$; then we let $z = 0^{\sqrt{N}/2}$ first, and set $z_a = i$ in $\mathcal{D}_n^i$; finally we output the pair (f,z) as a sample.

Distinguishing $\mathcal{D}_n^0$ and $\mathcal{D}_n^1$ is hard for SZK. Recall that SZK is a semantic class. That is, a given protocol Π might be invalid with different oracles or different inputs (i.e., the protocol might not satisfy the zero-knowledge constraint, or the verifier might accept with a probability that is neither $\geq 2/3$ nor $\leq 1/3$). We write $\Pi^{(f,z)}() = \perp$ when Π is invalid given oracle access to (f,z).

We claim that for any protocol Π, one of the following two claims must hold for sufficiently large n:

(A) $\Pr_{(f,z)\leftarrow\mathcal{D}_n^0}\left[\Pi^{(f,z)}() = \perp\right] > 0.1$ or $\Pr_{(f,z)\leftarrow\mathcal{D}_n^1}\left[\Pi^{(f,z)}() = \perp\right] > 0.1$.

(B) $\left|\Pr_{(f,z)\leftarrow\mathcal{D}_n^0}\left[\Pi^{(f,z)}() = 1\right] - \Pr_{(f,z)\leftarrow\mathcal{D}_n^1}\left[\Pi^{(f,z)}() = 1\right]\right| < 0.2$.

That is, either Π is invalid on a large fraction of oracles, or else Π cannot distinguish $\mathcal{D}_n^0$ from $\mathcal{D}_n^1$ with a very good probability.

Building a BPP algorithm to break $\mathsf{PRF}^{\mathsf{mod}}$. Suppose for a contradiction that there are infinitely many n such that none of (A) and (B) hold. Without loss of generality, we can assume that

$$\Pr_{(f,z)\leftarrow\mathcal{D}_n^1}\left[\Pi^{(f,z)}() = 1\right] - \Pr_{(f,z)\leftarrow\mathcal{D}_n^0}\left[\Pi^{(f,z)}() = 1\right] \geq 0.2.$$

We are going to build a BPP algorithm which is able to break $\mathsf{PRF}^{\mathsf{mod}}$ on those n, thereby contradicting Lemma 38.

From (A), we have

$$\Pr_{(f,z)\leftarrow\mathcal{D}_n^1}\left[\Pi^{(f,z)}() = 1\right] - \left(1 - \Pr_{(f,z)\leftarrow\mathcal{D}_n^0}\left[\Pi^{(f,z)}() = 0\right]\right) \geq 0.1,$$

which simplifies to

$$\Pr_{(f,z)\leftarrow\mathcal{D}_n^1}\left[\Pi^{(f,z)}() = 1\right] + \Pr_{(f,z)\leftarrow\mathcal{D}_n^0}\left[\Pi^{(f,z)}() = 0\right] \geq 1.1.$$

From the definition of $\mathcal{D}_n^0$ and $\mathcal{D}_n^1$, the above implies that

$$\Pr_{(k,a)\leftarrow\mathcal{K}^{\mathsf{mod}}}\left[\Pi^{(f,z_1)}() = 1 \text{ and } \Pi^{(f,z_0)}() = 0, f = \mathsf{PRF}^{\mathsf{mod}}_{(k,a)}, z_0 = 0^{\sqrt{N}/2}, z_1 = e_a\right] \geq 0.1,$$

where e_a denotes the string of length $\sqrt{N}/2$ that is all zero except for the a-th bit.

Analysis of distributions $A_i^{(f,z)}$. By a result of Sahai and Vadhan [51], there are two polynomial-time samplable distributions $A_0^{(f,z)}$ and $A_1^{(f,z)}$ such that $\|A_0^{(f,z)} - A_1^{(f,z)}\| \geq 1 - 2^{-n}$ when $\Pi^{(f,z)}() = 1$; and $\|A_0^{(f,z)} - A_1^{(f,z)}\| \leq 2^{-n}$ when $\Pi^{(f,z)}() = 0$.

Hence, with probability 0.1 over $(k,a) \leftarrow \mathcal{K}^{\mathsf{mod}}$, we have

$$\|A_0^{(f,z_1)} - A_1^{(f,z_1)}\| \geq 1 - 2^{-n} \text{ and } \|A_0^{(f,z_0)} - A_1^{(f,z_0)}\| \leq 2^{-n}.$$

This means that either $\|A_0^{(f,z_0)} - A_0^{(f,z_1)}\| \geq 1/3$ or $\|A_1^{(f,z_0)} - A_1^{(f,z_1)}\| \geq 1/3$.

Now we show that the above implies an algorithm that breaks $\mathsf{PRF}^{\mathsf{mod}}$, and therefore contradicts Lemma 38.

The algorithm and its analysis. Given oracle access to a function $f \leftarrow \mathsf{PRF}^{\mathsf{mod}}_{\mathcal{K}^{\mathsf{mod}}}$, our algorithm first picks a random index $i \in \{0,1\}$. It then simulates A_i with oracle access to (f,z) to take a sample from $A_i^{(f,z)}$, where $z = z_0 = 0^{\sqrt{N}/2}$; it records all the indices in z that are queried by A_i. Now, with probability at least $0.1/2 = 0.05$, we have $\|A_i^{(f,z_0)} - A_i^{(f,z_1)}\| \geq 1/3$. Since (f,z_0) and (f,z_1) only differ at the a-th index of z, we can see that $A_i^{(f,z_0)}$ must have queried the a-th index of z with probability at least $1/3$.

Hence, with probability at least $0.05/3 = \Omega(1)$, one of the values recorded by our algorithm is a, and in that case our algorithm can find a collision in f easily. However, when f is a truly random function, no algorithm can find a collision with a non-negligible probability. Therefore, this algorithm is a distinguisher between $\mathsf{PRF}^{\mathsf{mod}}$ and a truly random function, contradicting the fact that $\mathsf{PRF}^{\mathsf{mod}}$ is secure by Lemma 38.

Construction of the oracle $\mathcal{O}$. Finally, we are ready to construct our oracle $\mathcal{O}$. We will let $\mathcal{O}$ encode pairs $(f_1,z_1),(f_2,z_2),dotsc$, where f_n is a function from $\mathcal{X}_n$ to $\mathcal{X}_n$ and $z_n \in \{0,1\}^{\sqrt{N}/2}$.

For each n, we draw a random index $i \leftarrow \{0,1\}$, and then draw $(f_n,z_n) \leftarrow \mathcal{D}_n^i$. We set L to be the unary language consisting of all 0^n for which (f_n,z_n) is drawn from $\mathcal{D}_n^1$.

From Lemma 38, a quantum algorithm can distinguish $\mathcal{D}_n^0$ from $\mathcal{D}_n^1$, except with negligible probability, by recovering a. Therefore, by a similar argument as in the proof of Theorem 39, we have $L \in \mathsf{BQP}^{\mathcal{O}}$ with probability 1.

On the other hand, for a protocol Π and a sufficiently large n, either (A) happens, which means that $\Pi^{(f_n,z_n)}$ is invalid with probability 0.05 on input 0^n, or (B) happens, which means that Π cannot distinguish $\mathcal{D}_n^0$ and $\mathcal{D}_n^1$ with a constant probability.

In both cases Π cannot decide whether 0^n belongs to L correctly with bounded error. Hence, again by a similar argument as in the proof of Theorem 39, the probability that Π decides L is 0. And since there are only countably many protocols, we have $L \notin \mathsf{SZK}^{\mathcal{O}}$ with probability 1, which means that $\mathsf{BQP}^{\mathcal{O}} \not\subset \mathsf{SZK}^{\mathcal{O}}$ with probability 1.

Finally, it is easy to see that $\mathcal{O} \in \mathsf{P/poly}$, which completes the proof. ◀

B Missing Proofs in Section 3

We first prove Lemma 13.

Proof of Lemma 13. Let $N = 2^n$ for simplicity and L be a list consisting of N reals: $|\langle u|w\rangle|^2 - 2^{-n}$ for each $w \in \{0,1\}^n$. We sort all reals in L in increasing order, and denote them by $a_1, a_2, dotsc, a_N$. We also let $\Delta = \mathsf{dev}(|u\rangle)$ for brevity.

Then from the definitions of $\mathsf{adv}(|u\rangle)$ and $\mathsf{dev}(|u\rangle)$, we have

$$\sum_{i=1}^{N} a_i = 0,$$

$$\sum_{i=1}^{N} |a_i| = \Delta, \text{ and}$$

$$\mathsf{adv}(|u\rangle) = \frac{1}{2} + \sum_{i=N/2+1}^{N} a_i.$$

Now, let t be the first index such that $a_t \geq 0$. Then we have

$$\sum_{i=t}^{N} a_i = \sum_{i=t}^{N} |a_i| = \frac{\Delta}{2} \qquad \text{and} \qquad \sum_{i=1}^{t-1} a_i = -\sum_{i=1}^{t-1} |a_i| = -\frac{\Delta}{2}.$$

We are going to consider the following two cases.

(i) $t \geq N/2 + 1$. Note that a_i's are increasing and for all $i < t$, $a_i < 0$, we have

$$\sum_{i=1}^{N/2} |a_i| \geq \sum_{i=N/2+1}^{t-1} |a_i|,$$

which means

$$\sum_{i=N/2+1}^{t-1} |a_i| \leq \frac{1}{2} \cdot \sum_{i=1}^{t-1} |a_i| \leq \frac{\Delta}{4}.$$

Therefore,

$$\sum_{i=N/2+1}^{N} a_i \geq \sum_{i=t}^{N} a_i + \sum_{i=N/2+1}^{t-1} a_i \geq \frac{1}{2} + \frac{\Delta}{2} - \frac{\Delta}{4} \geq \frac{1}{2} + \frac{\Delta}{4}.$$

(ii) $t \leq N/2$. In this case, note that we have

$$\sum_{i=N/2+1}^{N} a_i \geq \sum_{i=t}^{N/2} a_i.$$

Therefore,

$$\sum_{i=N/2+1}^{N} a_i \geq \frac{1}{2} \cdot \sum_{i=t}^{N} a_i \geq \frac{\Delta}{4}.$$

Since in both cases we have $\sum_{i=N/2+1}^{N} a_i \geq \frac{\Delta}{4}$, it follows that

$$\mathsf{adv}(|u\rangle) = \frac{1}{2} + \sum_{i=N/2+1}^{N} a_i \geq \frac{1}{2} + \frac{\Delta}{4},$$

which completes the proof. ◀

Now we prove Lemma 14.

Proof of Lemma 14. The random pure state $|u\rangle$ can be generated as follows: draw four i.i.d. reals $x_1, x_2, x_3, x_4 \sim \mathcal{N}(0,1)$, and set

$$|u\rangle = \frac{(x_1 + x_2 i)|0\rangle + (x_3 + x_4 i)|1\rangle}{\sqrt{x_1^2 + x_2^2 + x_3^2 + x_4^2}}.$$

Hence, we have

$$\begin{aligned}
&\mathbb{E}\left[\left||\langle u|0\rangle|^2-|\langle u|1\rangle|^2\right|\right]\\
=&\int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\frac{1}{(2\pi)^2}\frac{|x_1^2+x_2^2-x_3^2-x_4^2|}{x_1^2+x_2^2+x_3^2+x_4^2}\cdot e^{-(x_1^2+x_2^2+x_3^2+x_4^2)/2}dx_1dx_2dx_3dx_4\\
=&\int_0^{2\pi}\int_0^{2\pi}\int_0^{+\infty}\int_0^{+\infty}\frac{1}{(2\pi)^2}\cdot\frac{|\rho_1^2-\rho_2^2|}{\rho_1^2+\rho_2^2}\cdot\rho_1\rho_2\cdot e^{-(\rho_1^2+\rho_2^2)/2}d\rho_1d\rho_2d\theta_1d\theta_2\\
&\qquad\qquad(x_1=\rho_1\sin\theta_1,\ y_1=\rho_1\cos\theta_1,\ x_2=\rho_2\sin\theta_2,\ y_2=\rho_2\cos\theta_2)\\
=&\int_0^{+\infty}\int_0^{+\infty}\frac{|\rho_1^2-\rho_2^2|}{\rho_1^2+\rho_2^2}\cdot\rho_1\rho_2\cdot e^{-(\rho_1^2+\rho_2^2)/2}d\rho_1d\rho_2\\
=&\frac{1}{2}
\end{aligned}$$

◀

C Missing Proofs in Section 6

We prove Lemma 30 here.

Proof of Lemma 30. We prove the concentration inequality by bounding the variance,

$$\mathrm{Var}[\mathsf{adv}(f)]=\mathbb{E}[\mathsf{adv}(f)^2]-\mathbb{E}[\mathsf{adv}(f)]^2.$$

Note that

$$\mathbb{E}[\mathsf{adv}(f)]^2=\left(\frac{2}{\sqrt{2\pi}}\int_1^{+\infty}x^2e^{-x^2/2}dx\right)^2=\mathsf{Succ}_Q^2.$$

We now calculate $\mathbb{E}[\mathsf{adv}(f)^2]$. We have

$$\begin{aligned}
\mathop{\mathbb{E}}_f[\mathsf{adv}(f)^2]&=\mathop{\mathbb{E}}_f\left[\left(\mathop{\mathbb{E}}_{z\in\{0,1\}^n}[\widehat{f}^2(z)\cdot\mathbf{1}_{|\widehat{f}(z)|\geq 1}]\right)^2\right]\\
&=\mathop{\mathbb{E}}_f\left[\mathop{\mathbb{E}}_{z_1,z_2\in\{0,1\}^n}\left[\widehat{f}^2(z_1)\widehat{f}^2(z_2)\cdot\mathbf{1}_{|\widehat{f}(z_1)|\geq 1\wedge|\widehat{f}(z_2)|\geq 1}\right]\right].\\
&=\mathop{\mathbb{E}}_{z_1,z_2\in\{0,1\}^n}\left[\mathop{\mathbb{E}}_f\left[\widehat{f}^2(z_1)\widehat{f}^2(z_2)\cdot\mathbf{1}_{|\widehat{f}(z_1)|\geq 1\wedge|\widehat{f}(z_2)|\geq 1}\right]\right].
\end{aligned}$$

Now there are two cases: $z_1=z_2$ and $z_1\neq z_2$. When $z_1=z_2$, let $z=z_1=z_2$; then we have

$$\begin{aligned}
Ex_f\left[\widehat{f}^2(z_1)\widehat{f}^2(z_2)\cdot\mathbf{1}_{|\widehat{f}(z_1)|\geq 1\wedge|\widehat{f}(z_2)|\geq 1}\right]&=\mathop{\mathbb{E}}_f\left[\widehat{f}^4(z)\cdot\mathbf{1}_{|\widehat{f}(z)|\geq 1}\right]\\
&=\frac{2}{\sqrt{2\pi}}\int_1^{+\infty}x^4e^{-x^2/2}dx\\
&=O(1).
\end{aligned}$$

Next, if $z_1\neq z_2$, then without loss of generality, we can assume $z_1=0^N$. Now we define two sets A and B,

$$A=\{x\in\{0,1\}^n:(z_2\cdot x)=0\}\text{ and }B=\{x\in\{0,1\}^n:(z_2\cdot x)=1\}.$$

We also define

$$\widehat{f}_A:=\frac{1}{\sqrt{N/2}}\cdot\sum_{z\in A}f(z)\text{ and }\widehat{f}_B:=\frac{1}{\sqrt{N/2}}\cdot\sum_{z\in B}f(z).$$

Then from the definitions of $\widehat{f}(z_1)$ and $\widehat{f}(z_2)$, we have

$$\widehat{f}(z_1) = \frac{1}{\sqrt{2}} \cdot (\widehat{f}_A + \widehat{f}_B) \text{ and } \widehat{f}(z_1) = \frac{1}{\sqrt{2}} \cdot (\widehat{f}_A - \widehat{f}_B).$$

Therefore,

$$\mathop{\mathbb{E}}_{f}\left[\widehat{f}^2(z_1)\widehat{f}^2(z_2) \cdot \mathbf{1}_{|\widehat{f}(z_1)|\geq 1 \wedge |\widehat{f}(z_2)|\geq 1}\right] = \left(\frac{1}{\sqrt{2\pi}}\right)^2 \cdot \int_{\substack{|a+b|\geq\sqrt{2} \\ |a-b|\geq\sqrt{2}}} \frac{1}{4} \cdot (a+b)^2 \cdot (a-b)^2 \cdot e^{-(a^2+b^2)/2} \cdot dadb.$$

Let $x = a + b$ and $y = a - b$. Then

$$a = \frac{x+y}{2}, \quad b = \frac{x-y}{2}, \quad da = \frac{dx+dy}{2}, \quad \text{and} \quad db = \frac{dx-dy}{2}.$$

Also note that $x^2 + y^2 = 2(a^2 + b^2)$. Plugging in x and y, the above can be simplified to

$$\begin{aligned}
\frac{1}{2\pi}\int_{\substack{|x|\geq\sqrt{2} \\ |y|\geq\sqrt{2}}} \frac{1}{4}x^2y^2e^{-(x^2+y^2)/4} \cdot \frac{1}{2}dxdy &= \frac{1}{2\pi}\left(\int_{|x|\geq\sqrt{2}} \frac{1}{2\sqrt{2}} \cdot x^2e^{-x^2/4}dx\right)^2 \\
&= \frac{1}{2\pi}\left(\int_{\sqrt{2}}^{+\infty} \frac{1}{\sqrt{2}} \cdot x^2e^{-x^2/4}dx\right)^2 \\
&= \frac{1}{2\pi}\left(\int_{1}^{+\infty} 2t^2e^{-t^2/2}dt\right)^2 && (t = x/\sqrt{2}) \\
&= \left(\frac{2}{\sqrt{2\pi}}\int_{1}^{+\infty} t^2e^{-t^2/2}dt\right)^2 = && \mathsf{Succ}_Q^2.
\end{aligned}$$

Putting two cases together, we have

$$\mathop{\mathbb{E}}_{f}[\mathsf{adv}(f)^2] = \frac{1}{N} \cdot O(1) + \frac{N-1}{N} \cdot \mathsf{Succ}_Q^2,$$

which in turn implies

$$\mathrm{Var}[\mathsf{adv}(f)] = O(1/N).$$ ◀

D Missing Proofs in Section 7

For completeness, we prove Lemma 38 here.

Proof of Lemma 38. In the following, we will always use $\varepsilon = \varepsilon(n)$ to denote a negligible function. And we will denote $\mathcal{X}^{\mathsf{raw}}$ as $\mathcal{X}$ for brevity. Recall that we interpret $\mathcal{X}$ as $[N]$ for $N = N(n) = \mathcal{X}$.

Both $\mathsf{PRP}^{\mathsf{raw}}$ and $\mathsf{PRF}^{\mathsf{mod}}$ are classically-secure PRFs. It is well-known that a secure PRP is also a secure PRF; therefore $\mathsf{PRP}^{\mathsf{raw}}$ is a classically-secure PRF. So we only need to prove this for $\mathsf{PRF}^{\mathsf{mod}}$.

Recall that $\mathsf{PRF}^{\mathsf{mod}}_{(k,a)}(x) = \mathsf{PRP}^{\mathsf{raw}}_k((x-1) \bmod a + 1)$. We first show that if the $\mathsf{PRP}^{\mathsf{raw}}$ in the definition of $\mathsf{PRF}^{\mathsf{mod}}$ were replaced by a truly random function, then no classical polynomial-time algorithm A could distinguish it from a truly random function. That is,

$$\left|\Pr_{f\leftarrow\mathcal{X}^{\mathcal{X}},a\leftarrow\mathcal{A}}[A^{f\bmod a}() = 1] - \Pr_{f\leftarrow\mathcal{X}^{\mathcal{X}}}[A^{f}() = 1]\right| < \varepsilon, \tag{23}$$

where $f_{\bmod a}(x) := f((x-1) \bmod a + 1)$.

Clearly, as long as A never queries its oracle on two points x and x' such that $x \equiv x' \pmod{a}$, the oracle will look random. Suppose A makes q queries in total. There are $\binom{q}{2}$ possible differences between query points, and each difference is at most N. So for large enough N, each difference can be divisible by at most two different moduli from $\mathcal{A}$ (recall that each number in $\mathcal{A}$ lies in $[\sqrt{N}/4, sqrtN/2]$). And since $|\mathcal{A}| \geq \Omega(\sqrt{N}/\log N)$, the total probability of querying two x and x' such that $x \equiv x' \pmod{a}$ is at most

$$O\left(\frac{q^2 \log N}{\sqrt{N}}\right),$$

which is negligible as N is exponential in n. This implies (23).

Now, since $\mathsf{PRP}^{\mathsf{raw}}$ is a classically-secure PRF, for any polynomial-time algorithm A, we have

$$\left| \Pr_{f \leftarrow \mathcal{X}^{\mathcal{X}}, a \leftarrow \mathcal{A}} [A^{f \bmod a}() = 1] - \Pr_{f \leftarrow \mathsf{PRP}^{\mathsf{raw}}_{\mathcal{K}^{\mathsf{raw}}}, a \leftarrow \mathcal{A}} [A^{f \bmod a}() = 1] \right| < \varepsilon, \tag{24}$$

since otherwise we can directly construct a distinguisher between $\mathsf{PRP}^{\mathsf{raw}}_{\mathcal{K}^{\mathsf{raw}}}$ and $\mathcal{X}^{\mathcal{X}}$.

Note that

$$\Pr_{f \leftarrow \mathsf{PRP}^{\mathsf{raw}}_{\mathcal{K}^{\mathsf{raw}}}, a \leftarrow \mathcal{A}} [A^{f \bmod a}() = 1] = \Pr_{f \leftarrow \mathsf{PRF}^{\mathsf{mod}}_{\mathcal{K}^{\mathsf{mod}}}} [A^{f}() = 1]$$

by their definitions. Hence, (23) and (24) together imply that

$$\left| \Pr_{f \leftarrow \mathsf{PRF}^{\mathsf{mod}}_{\mathcal{K}^{\mathsf{mod}}}} [A^{f}() = 1] - \Pr_{f \leftarrow \mathcal{X}^{\mathcal{X}}} [A^{f}() = 1] \right| < \varepsilon$$

for any polynomial-time algorithm A. This completes the proof for the first statement.

Quantum algorithm for recovering a given oracle access to $\mathsf{PRF}^{\mathsf{mod}}_{\mathcal{K}^{\mathsf{mod}}}$. Let $(k, a) \leftarrow \mathcal{K}^{\mathsf{mod}}$, $f = \mathsf{PRF}^{\mathsf{mod}}_{(k,a)}$ and $g = \mathsf{PRP}^{\mathsf{raw}}_{k}$. From the definitions, we have $f = g \bmod a$.

Since g is a permutation, there is no collision (x, x') such that $g(x) = g(x')$. Moreover, in this case, $f = g \bmod a$ has a unique period a. Therefore, we can apply Boneh and Lipton's quantum period-finding algorithm [20] to recover a. Using a polynomial number of repetitions, we can make the failure probability negligible, which completes the proof for the second statement.

Quantum algorithm for distinguishing $\mathsf{PRP}^{\mathsf{raw}}$ and $\mathsf{PRF}^{\mathsf{mod}}$. Finally, we show the above algorithm implies a good quantum distinguisher between $\mathsf{PRP}^{\mathsf{raw}}$ and $\mathsf{PRF}^{\mathsf{mod}}$. Given oracle access to a function f, our distinguisher A tries to recover a period a using the previously discussed algorithm, and accepts only if $f(1) = f(1 + a)$.

When $f \leftarrow \mathsf{PRP}^{\mathsf{raw}}_{\mathcal{K}^{\mathsf{raw}}}$, note that f is a permutation, which means A accepts with probability 0 in this case.

On the other side, when $f \leftarrow \mathsf{PRF}^{\mathsf{mod}}_{\mathcal{K}^{\mathsf{mod}}}$, from the second statement, A can recover the period a with probability at least $1 - \varepsilon$. Therefore A accepts with probability at least $1 - \varepsilon$.

Combining, we find that A is a distinguisher with advantage $1 - \varepsilon$, and this completes the proof for the last statement. ◀

E Numerical Simulation For Conjecture 1

Recall Conjecture 1, which said that a random quantum circuit C on n qubits satisfies $\mathsf{adv}(C) \geq C_{\mathsf{thr}} - \varepsilon$ with probability $1 - 1/\exp(n)$, where

$$C_{\mathsf{thr}} := \frac{1+\ln 2}{2}.$$

We first explain where the magic number C_{thr} comes from. Suppose C is drawn from $\mu_{\mathsf{Haar}}^{2^n}$ instead of μ_{grid}. Then $C|0^n\rangle$ is a random quantum state, and therefore the values $2^n \cdot |\langle x|C|0\rangle|$'s, for each $x \in \{0,1\}^n$ are distributed very closely to 2^n i.i.d. exponential distributions with $\lambda = 1$.

So, assuming that, we can see that the median of $\mathsf{probList}(C|0\rangle)$ concentrates around $\ln 2$, as

$$\int_0^{\ln 2} e^{-x} dx = \frac{1}{2},$$

which also implies that $\mathsf{adv}(C)$ concentrates around

$$\int_{\ln 2}^{+\infty} x e^{-x} dx = C_{\mathsf{thr}} = \frac{1+\ln 2}{2} \approx 0.846574.$$

In the following, we first provide some numerical evidence that the values in $\mathsf{probList}(C|0\rangle)$ *also* behave like exponentially distributed random variables, which explains why the constant should indeed be C_{thr}. Then we provide a direct numerical simulation for the distribution of $\mathsf{adv}(C)$ to argue that $\mathsf{adv}(C)$ approximately follows a nice normal distribution. Finally we examine the decreasing rate of the standard variance of $\mathsf{adv}(C)$ to support our conjecture.

E.1 Numerical Simulation Setting

In the following we usually set $n = 9$ or $n = 16$ (so that $\sqrt{n}$ is an integer); and we always set $m = n^2$ as in Conjecture 1.

E.2 Distribution of probList($C|0\rangle$): Approximate Exponential Distribution

In Figure 2 we plot the histogram of the distribution of the normalized probabilities in $\mathsf{probList}(C|0\rangle)$ where $C \leftarrow \mu_{\mathsf{grid}}^{16,256}$, that is,

$$\{2^n \cdot p : p \in \mathsf{probList}(C|0\rangle)\}.$$

And we compare it with the exponential distribution with $\lambda = 1$. From Figure 2, it is easy to observe that these two distributions are quite similar.

E.3 Distribution of adv(C): Approximate Normal Distribution

Next we perform direct numerical simulation to see how $\mathsf{adv}(C)$ is distributed when $C \leftarrow \mu_{\mathsf{grid}}^{n,m}$. Our results suggest that $\mathsf{adv}(C)$ approximately follows a normal distribution with mean close to C_{thr}.

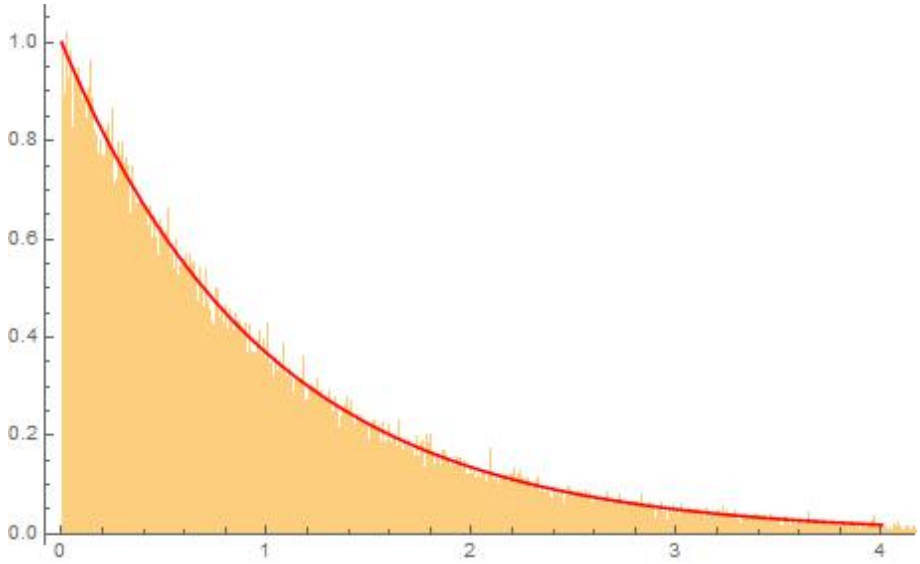

Figure 2 A histogram of (normalized) $\mathsf{probList}(C|0\rangle)$, where $C \leftarrow \mu_{\mathsf{grid}}^{16,256}$. The x-axis represents the probability, and the y-axis represents the estimated density, and the red line indicates the PDF of the exponential distribution with $\lambda = 1$.

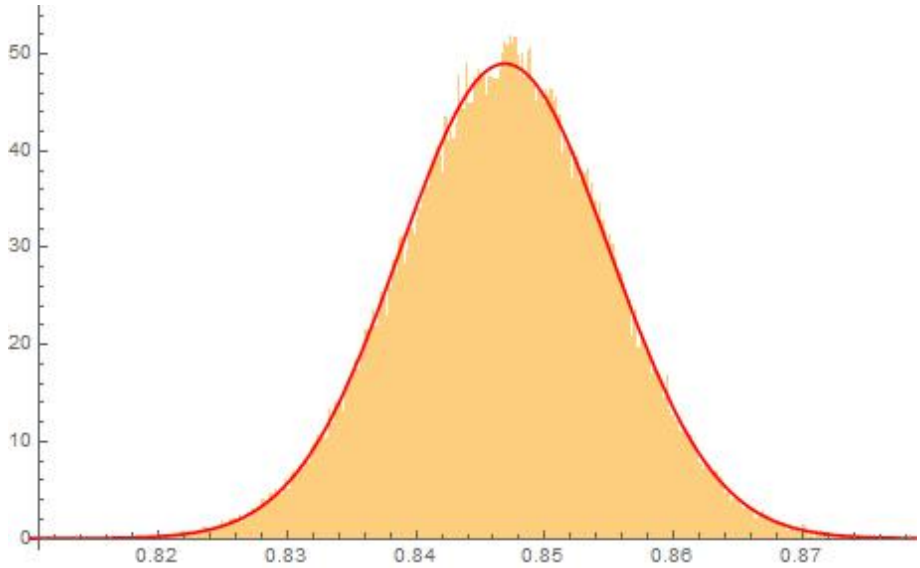

Figure 3 A histogram of the $\mathsf{adv}(C)$'s of the 10^5 i.i.d. samples from $\mu_{\mathsf{grid}}^{9,81}$. The x-axis represents the value of $\mathsf{adv}(C)$, and the y-axis represents the estimated density, and the red line indicates the PDF of the normal distribution $\mathcal{N}(0.846884, 0.00813911^2)$.

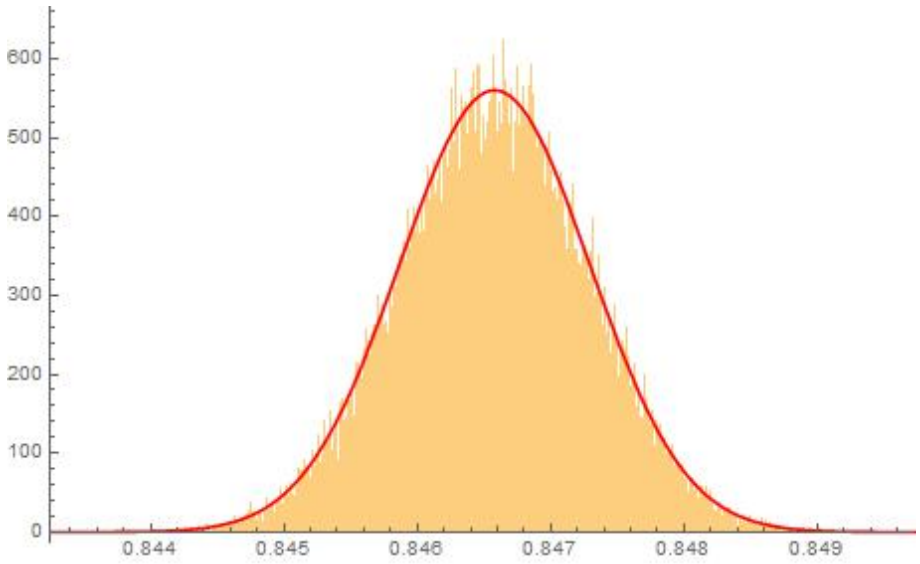

Figure 4 A histogram of the $\mathsf{adv}(C)$'s of the 10^5 i.i.d. samples from $\mu_{\mathsf{grid}}^{16,256}$. The x-axis represents the value of $\mathsf{adv}(C)$, the y-axis represents the estimated density, and the red line indicates the PDF of the normal distribution $\mathcal{N}(0.846579, 0.000712571^2)$.

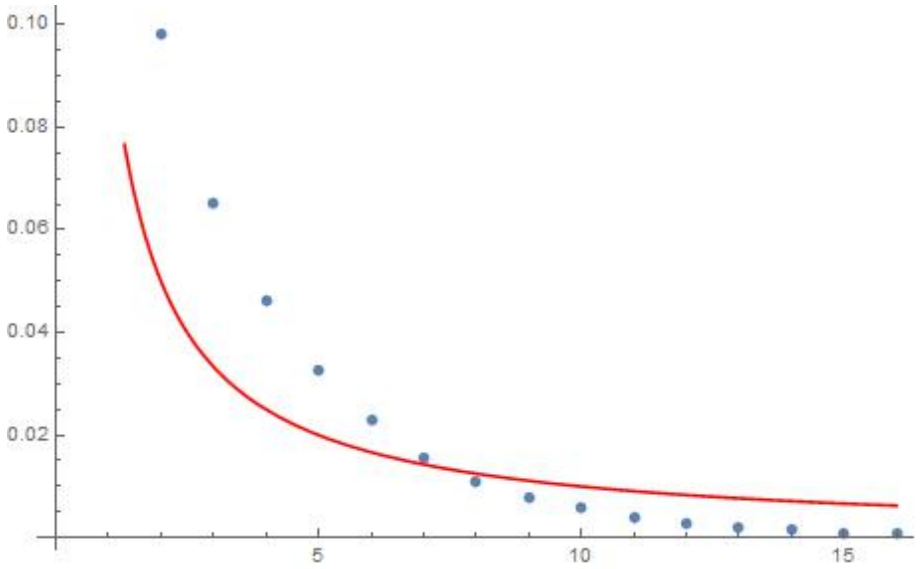

Figure 5 The empirical decay of the variance of $\mathsf{adv}(C)$. Here a point (x, y) means that the standard variance of the corresponding $\mathsf{adv}(C)$'s for the 1000 i.i.d. samples from $\mu_{\mathsf{general}}^{x,x^2}$ is y. Also, the red line represents the function $y = 0.1/x$.

E.3.1 $\mu_{\mathsf{grid}}^{9,81}$, 10^5 samples

We first draw 10^5 i.i.d. samples from $\mu_{\mathsf{grid}}^{9,81}$ and plot the distribution of the corresponding $\mathsf{adv}(C)$'s in Figure 3. From Figure 3, we can see that the distribution of $\mathsf{adv}(C)$ follows a nice normal distribution, with mean very close to C_{thr}.

E.3.2 $\mu_{\mathsf{grid}}^{16,256}$, 10^5 samples

Next, we draw 10^5 i.i.d. samples from $\mu_{\mathsf{grid}}^{16,256}$ and plot the distribution of the corresponding $\mathsf{adv}(C)$'s in Figure 4. From Figure 4, we can observe that the distribution of $\mathsf{adv}(C)$ in this case also mimics a nice normal distribution, with mean even closer to C_{thr} than in the previous case.

E.4 The Empirical Decay of Variance

The previous subsection suggests that $\mathsf{adv}(C)$ follows a normal distribution with mean approaching C_{thr}. If that's indeed the case, then informally, Conjecture 1 becomes equivalent to the conjecture that the variance σ of C_{thr} becomes $O(1/n)$ as $n \to +\infty$. So we wish to verify the latter conjecture for $\mu_{\mathsf{grid}}^{n,n^2}$ with some numerical simulation.

The circuit distribution $\mu_{\mathsf{general}}^{n,m}$

Unfortunately, the definition of $\mu_{\mathsf{grid}}^{n,m}$ requires n to be a perfect square, and there are only five perfect squares for which we can perform quick simulations ($n \in \{1, 4, 9, 16, 25\}$). So we consider the following distribution $\mu_{\mathsf{general}}^{n,m}$ on n qubits and m circuits instead: each of m gates is a Haar random two-qubit gate acting on two qubits chosen uniformly at random. In this case, since we don't need to arrange the qubits in a square grid, n can be any positive integer.

Numerical simulation shows that $\mathsf{adv}(C)$ is distributed nearly the same when C is drawn from $\mu_{\mathsf{general}}^{n,n^2}$ or $\mu_{\mathsf{grid}}^{n,n^2}$ for $n = 3$ or $n = 4$, so it is reasonable to consider μ_{general} instead of μ_{grid}.

For each $n = 2, 3, dotsc, 16$, we draw 1000 i.i.d. samples from $\mu_{\mathsf{general}}^{n,n^2}$, and calculate the variance of the corresponding $\mathsf{adv}(C)$'s. The results are summarized in Figure 5.

From Figure 5, we can observe that the variance decreases faster than the inverse function $1/x$; hence it supports Conjecture 1.