Quantum Algorithms in a Superposition of Spacetimes

Omri Shmueli*

Abstract

Quantum computers are expected to revolutionize our ability to process information. The advancement from classical to quantum computing is a product of our advancement from classical to quantum physics – the more our understanding of the universe grows, so does our ability to use it for computation. A natural question that arises is, what will physics allow in the future? Can more advanced theories of physics increase our computational power, beyond quantum computing?

An active field of research in physics studies theoretical phenomena outside the scope of explainable quantum mechanics, that form when attempting to combine Quantum Mechanics (QM) with General Relativity (GR) into a unified theory of Quantum Gravity (QG). One expected phenomenon in QG is quantum uncertainty in the structure of spacetime: QM and GR together may imply that spacetime can be in superposition of curvatures. Under GR, gravitational time dilation asserts that the geometry of spacetime determines the order of events occurring. Accordingly, a superposition of curvatures is known to present the possibility of a superposition of event orderings.

In the literature of quantum information theory, the most natural model for a superposition of event orders, is a superposition of unitary evolution orders: For unitaries U_0, U_1 and state $|\psi\rangle$, execution in a superposition of orders aims to generate the normalization of the state $U_1 U_0 \cdot |\psi\rangle + U_0 U_1 \cdot |\psi\rangle$. Costa (Quantum 2022) proves that a model with such power cannot produce a valid process matrix (a generalization of standard quantum processes), and leaves the analysis of any relaxation of the model as an open question.

In this work we show a first example of a natural computational model based on QG, that provides an exponential speedup over standard quantum computation (under standard hardness assumptions). Formally, we define a relaxation of the previous model of superposition of unitary orders, where we do allow the generation of the state $U_1 U_0 \cdot |\psi\rangle + U_0 U_1 \cdot |\psi\rangle$, but with computational complexity inversely proportional to the state's norm (unlike the previous model, considering unconditional generation). We further provide physical intuition and assumptions behind our model.

We show that a quantum computer with the ability to create a superposition of unitary orders is able to solve in polynomial time two of the fundamental problems in computer science: The Graph Isomorphism Problem (GI) and the Gap Closest Vector Problem (GapCVP), with gap $O(n^2)$. These problems are believed by experts to be hard to solve for a regular quantum computer. Interestingly, our model does not seem overpowered, and we found no obvious way to solve entire complexity classes that are considered hard in computer science, like the classes **NP** and **SZK**.

Contents

1   Introduction

Theoretical physics is in a quest to draw a blueprint for the universe. The constructed different theories, which are sets of rules postulating how the universe works in certain scenarios, are our tool for predicting what will be the next step of a physical process. The two most accurate theories to date are Quantum Mechanics (QM) and General Relativity (GR). QM (along with Quantum Field Theory) describes the interaction between three out of the four fundamental forces in nature: Strong nuclear force, weak nuclear force and electromagnetism. GR describes the fourth force – gravity. Both QM and GR are immensely successful theories, in understanding vastly different physical phenomena.

Formulating an experimentally-verified theory of physics that generalizes both QM and GR, is one of the grand challenges of physics. Such a theory, usually referred to as a theory of Quantum Gravity, will model the interaction between quantum effects and gravitational force; an interaction which is currently ignored in most analyses. The two most popular candidates as theories of quantum gravity are String Theory and Loop Quantum Gravity, however, numerous other proposals exist [dBDE$^+$22].

Moving from physics to computer science, we observe an interconnectedness. On the one hand, theories of physics dictate the potential computational power in humanity's hand. If a theory of physics allows for a machine with new information processing capabilities, this sometimes implies algorithmic breakthroughs: For example, quantum mechanics allows for a quantum computer, which in turn is needed to run Shor's algorithm [Sho99]. Shor's algorithm factors integers in polynomial time, whereas our fastest (known) factoring algorithms on any classical computer run in sub-exponential time. In practical terms, the meaning of these timescales is the difference between possible and impossible.

On the other hand, computational insights shed light back on physics. To see this, consider two physical theories A and B, where B claims to be a more elaborate description of the universe and generalize theory A. In that case, B can not only explain all physical processes that A can, but can accurately describe processes which A cannot. There is a subtlety here; While we can encounter a physical process that theory A *currently* does not know how to explain, it does not necessarily mean that there is no explanation – it might exist, and we just haven't found it yet. Proving that there is no explanation, however, is a task of different magnitude. Turning to computer science for help, we can construct two computational models corresponding to the two theories (i.e., a strongest computer C_A that can be constructed according to theory A and a computer C_B that can be constructed according to theory B). Now, if we find a computational problem P that C_B can solve but C_A cannot, this gives strong evidence that theory B can explain physical phenomena that A cannot, namely, the solving of P! Given the correspondence between algorithms and theoretical physics, a natural question which arises is the following: Just like the progression from classical to quantum mechanics allowed computational transcendence, how can a theory of quantum gravity transform our theory of computation?

Superposition of Spacetime Geometries. While a confirmed theory of quantum gravity remains elusive, there are some physical phenomena expected to take place within *any* valid quantum gravity theory, in some form or another. Arguably the most basic expected phenomenon unique to quantum gravity, is the possibility of quantum uncertainty in the structure spacetime itself [Ash96]. More elaborately, according to QM, all dynamic quantities are subject to quantum superposition. GR tells us that the geometry of spacetime is not fixed, and a straightforward combination implies the possibility for a superposition of spacetime geometries [Ash96, GCRB19, CR19, GB20, Gia21, GB22].

Structure of the Introduction. The goal of this paper is to suggest and define a computational model for a quantum computer with the added ability of generating a superposition of spacetime geometries, and to analyze its power. The rest of the Introduction is as follows. In Section 1.1 we provide a brief overview of the physics literature on a superposition of spacetime geometries, and then explain how it serves as inspiration for our computational model. This part deals with physics, it is self contained and can be skipped for the readers that are interested only in the complexity theoretical aspects of this work. In Section 1.2 we present our computational model and its connection to both standard quantum

computation and previous models in the quantum information theory. In Section 1.3 we present our complexity theoretical results for the model.

1.1 Gravity, Superposition and Entanglement

Before we explain what a quantum spacetime may be able to do, we refer to one ability of a *classical* spacetime. GR postulates the effects of gravitational time dilation: The structure of spacetime determines the pace at which time flows. More precisely, gravitational time dilation is the effect of time moving slower in a region of space with greater gravitational potential: For some location $s \in \mathbb{R}^3$ in space, the time in s slows down proportionally to how close s is to a massive object, and the mass of that object. According to this logic, a quantum spacetime may enable a superposition of spacetime curvatures, which in turn induces a superposition of time dilation effects, on the same set of events.

Gravitational Decoherence Versus Gravitationally-Induced Entanglement. Let us be more concrete, by using (with small variations) a thought experiment by Zych, Costa, Pikovski and Brukner [ZCPB19] (depicted in Figure 1 from [ZCPB19]). In the experiment there are two *spatially isolated* systems; one contains a gravitationally significant object M, and the second has two clocked unitary circuits U_0, U_1. More precisely,

- In system S_M we can generate a uniform superposition of two possible locations L_0, L_1 of a mass M. Note that we are using the fact that quantum theory does not prohibit superpositions of massive objects.

- In system S_U there is a quantum register R in some state $|\psi\rangle$, two atomic clocks C_0, C_1, and a central processing unit C_c that gets signals from the clocks. C_c always executes U_0 and U_1 on R, and the order in which it executes the two unitary circuits depends on the signal from which clock, C_0 or C_1, arrives first. For the sake of order, system S_U is constructed such that clock C_1 takes a bit more time (i.e., needs to make more clock ticks), so that when no disturbances are applied to system S_U, the order in which C_c executes the unitaries is always U_0 first and then U_1. Such a system should also be realizable by quantum mechanics.

- The last detail is the only one that uses effects from GR, specifically, gravitational time dilation. The system S_M is located with respect to the second system S_U such that, even though the systems are spatially disjoint in coordinates, if M is at location L_0, which is closer to clock C_1, then C_1 ticks slower and C_0 completes first, and if M is at location L_1, which is closer to C_0, then C_0 ticks slower and C_1 completes first.

Imagine we (1) execute S_M and generate a uniform superposition of the two possible locations for M, followed by (2) executing S_U, the clocked execution of the unitaries U_0, U_1. We can analyze what happens in each of the branches of the superposition, separately, assuming as if things are classical and there is no superposition. If M is at L_0, the state in R at the end of the execution is $U_1 U_0 |\psi\rangle$, and if M is at L_1 then the state in R at the end of execution is $U_0 U_1 |\psi\rangle$. The question becomes one in quantum gravity, when we ask what happens not in each branch of the superposition separately, but as a full joint system, in superposition. Experimental physics does not know the answer to this question.

The above question may translate to more basic questions about gravity, superposition and entanglement. Specifically, the first question is whether a massive object can be maintained in a quantum superposition, and the second question is whether entanglement can be created through gravity alone. More elaborately, on one side, the gravitational decoherence hypothesis [Diό89, Pen96, Dio14, Dio87] says that a quantum superposition of a massive object will fundamentally decohere, turning into a probabilistic mixture of classical states. On the other end, gravitationally-induced entanglement (GIE) [BMM+17, MV17] is hypothesizing that not only a massive object can be in a superposition, but that the gravitational field can be used to mediate entanglement.

In the context of the thought experiment from [ZCPB19], gravitational decoherence says that when executing the system S_M, there will not be a coherent superposition for the locations of M, so with probability $\frac{1}{2}$ the mass will be at L_0 and with the remaining probability at L_1. The end result of the execution of the two systems, in that case, is the probabilistic uniform mixture (or in other words, uniform mixed state) between

$$|L_0\rangle_{S_M} \otimes U_1 U_0 |\psi\rangle_{S_U} \ , \ |L_1\rangle_{S_M} \otimes U_0 U_1 |\psi\rangle_{S_U} \ ,$$

where for $b \in \{0, 1\}$, $|L_b\rangle$ is the state where the mass M is at location L_b, and we assume that the states of the different locations are orthogonal (that is, $\langle L_0 || L_1 \rangle = 0$). If GIE is true, then the per-branch analysis from before holds, only that we get the quantum state. Assuming GIE, at the end of the execution of the joint system, the quantum state is the pure state

$$\frac{1}{\sqrt{2}} \cdot |L_0\rangle_{S_M} \otimes U_1 U_0 |\psi\rangle_{S_U} + \frac{1}{\sqrt{2}} \cdot |L_1\rangle_{S_M} \otimes U_0 U_1 |\psi\rangle_{S_U} \ . \tag{1}$$

As mentioned earlier, using the effects of gravitational time dilation, gravity can be used to create correlations between two spatially isolated systems[1]. In the scope of this paper we call this phenomenon Gravitational Correlation. In the case of executing the experiment under gravitational decoherence, we can think of the result as a Classical Gravitational Correlation. In the case of executing the experiment under GIE, we consider the result as a Quantum Gravitational Correlation. Up to date and despite considerable efforts, no experiment exists that either proves or refutes gravitational decoherence or gravitationally-induced entanglement (that is, a reproducible experiment with statistical significance).

A complexity theoretical perspective. Both classical and quantum gravitational correlations are efficiently simulatable in the quantum circuit model. This is a basic exercise in quantum information processing: We require the information of the mass M (but not an actual massive body). By using controlled versions of the unitary circuits U_0, U_1, we can create the same state from the GIE execution 1. Then, if we want the state from the case of gravitational decoherence, we only need to measure the left register (containing the system S_M) in the computational basis.

Note that the two options of either a classical or quantum gravitational correlation (corresponding to gravitational decoherence or GIE, respectively), do not cover the entire field of possibilities for the outcome of the experiment. There is a possibility of quantum gravitational correlation *without entanglement*. That is, a massive object can be maintained in a coherent superposition, as opposed to gravitational decoherence, but gravity alone cannot create entanglement, as opposed to GIE. Indeed, some preliminary results suggest that even if spacetime is quantum, this does not necessarily implies it has to mediate entanglement [BWG+18, BWG+19, Ber19, DSW22, SMY23].

This work - quantum gravitational correlation without entanglement. The question we focus on in this research is, what are the computational implications of the the last scenario? Assuming quantum gravitational correlations without entanglement, what happens when the system S_U executes, while the system S_M finished executing and is in the state $\frac{1}{\sqrt{2}} \cdot |L_0\rangle_{S_M} + \frac{1}{\sqrt{2}} \cdot |L_1\rangle_{S_M}$? Our computational model makes the following assumptions. First, if quantum gravitational correlations are possible but cannot generate entanglement, when executing the experiment this should create a joint state that resembles the un-normalized state

$$(|L_0\rangle + |L_1\rangle)_{S_M} \otimes (U_1 U_0 |\psi\rangle + U_0 U_1 |\psi\rangle)_{S_U} \ .$$

We observe that the norm of the state $U_1 U_0 |\psi\rangle + U_0 U_1 |\psi\rangle$ fluctuates as a function of all components U_0, U_1 and $|\psi\rangle$. This means that (similarly to calculations stemming from general relativity in quantum gravity) such transformation will not be linear, due to the need for a state re-normalization. We imagine that in case that the physical universe allows to execute transformations $(U_0, U_1, |\psi\rangle) \rightarrow U_1 U_0 |\psi\rangle + U_0 U_1 |\psi\rangle$,

---

[1]Recall that this is exactly what happens in the above experiment – depending on the location of M in the system S_M, this determines a different spacetime geometry in the system S_U, which in turn cause the unitaries U_0, U_1 to be executed in one order or the other. This in particular creates a correlation between the states of the two systems.

it is numerically intuitive that decoherence will be inversely proportional to the norm of the state, and that energy could be invested to maintain coherence, which should also be inversely proportional to the norm. The above is the inspiration behind our computational model, which we present next.

1.2   Our Computational Model - Computable Order Interference

We now move from informal discussions in physics to a formal representation of our model. In the literature of quantum information theory, the phenomenon of uncertainty of the background spacetime is referred to as an indefinite causal structure (ICS) [Har07, Har05, MCH09, OCB12, CDPV13, LS13, Bru14]. As the structure of spacetime determines the order of events happening (but not necessarily the set of events happening), ICS studies the possibility of quantum uncertainty of event occurrence order. In quantum information theory, an event is the time-evolution of a physical system, or equivalently, the execution of a quantum circuit on a quantum register. Following these baselines, the most natural interpretation of ICS is a quantum superposition of unitary execution orders [OCB12, CDPV13]. Given two n-qubit unitary circuits U_0, U_1 and an n-qubit state $|\psi\rangle$, execution in ICS can be imagined as a superposition between two scenarios – one scenario where U_0 is executed followed by an execution of U_1, generating the state $U_1 U_0 \cdot |\psi\rangle$, and a second scenario with flipped execution order, generating $U_0 U_1 \cdot |\psi\rangle$. Executing U_0, U_1 on $|\psi\rangle$ with indefinite causal structure should attempt to create the state $U_1 U_0 \cdot |\psi\rangle + U_0 U_1 \cdot |\psi\rangle$.

There is a crucial difference between a *pure* superposition of unitary orders, which takes the above form $U_1 U_0 \cdot |\psi\rangle + U_0 U_1 \cdot |\psi\rangle$, and a mixed superposition, entangled with an auxiliary register, which takes the form $|0\rangle U_1 U_0 \cdot |\psi\rangle + |1\rangle U_0 U_1 \cdot |\psi\rangle$. First, the latter can be easily generated in the standard quantum circuit model, using controlled versions of U_0, U_1. Second, the former is generally not even a unitary transformation. More so, Costa [Cos22] shows that such transformation cannot produce a Process Matrix, which is, in a nutshell, a non-causally ordered generalization of quantum states [CDPV13]. To elaborate, the Process Matrix Formalism [CDPV13] (PMF) is a mathematical framework for defining quantum processes that are not constrained by a causal execution order. The PMF defines what is a valid process matrix W, and while the direct physical meaning of process matrices is currently not well understood[2], the formalism remains an important tool. In particular, as in the case of [Cos22], the formalism can be used to *exclude* some transformations from being considered valid quantum transformations, in the more broad sense that is free of causal restrictions (that is, a transformation which is not a process matrix is in particular not a valid quantum process in standard quantum mechanics). Costa leaves open the question of understanding the computational power of any relaxation of the model of superposition of unitary orders.

Execution of superposition of unitary orders as a function of interference norm. We start our work by exploring more carefully the idea of a superposition of unitary orders, where our goal is to define a natural relaxation of it. From hereon we call the state $U_1 U_0 |\psi\rangle + U_0 U_1 |\psi\rangle$ the *order interference* state, or OI state for short[3]. We can think of order superposition generalized to an arbitrary number $m \in \mathbb{N}$ of unitaries $\{U_i\}_{i \in [m]}$, and the OI state is defined as

$$\mathbf{OI}\left(\{U_i\}_{i \in [m]}, |\psi\rangle\right) := \sum_{\sigma \in S_m} \left(\prod_{i \in [m]} U_{\sigma^{-1}(i)}\right) \cdot |\psi\rangle \ , \tag{2}$$

---

[2]In particular, the process matrix formalism is not known to be connected to gravitational interaction or general relativity, and is written only in the language of quantum information processing.

[3]As we mentioned before, generating the state $|0\rangle U_1 U_0 \cdot |\psi\rangle + |1\rangle U_0 U_1 \cdot |\psi\rangle$, which can be thought of as a superposition of evolution orders, is easy in quantum mechanics. However, while there is superposition, there is no interference between different evolution orders. Unlike some of the previous work on the subject (e.g. Quantum Switch [CDPV13]), we view not the superposition, but the interference between different evolution orders as a possibly unique idea to quantum gravity.

where S_m is the set of all permutations on the set $[m] := \{1, 2, 3, \cdots, m\}$, and the above product notation $\prod_{i \in [m]}$ creates multiplications between matrices enumerated from right to left, which is also the convention for matrix multiplication in the rest of this work, that is:

$$\prod_{i \in [m]} U_{\sigma^{-1}(i)} := U_{\sigma^{-1}(m)} \cdot U_{\sigma^{-1}(m-1)} \cdots U_{\sigma^{-1}(2)} \cdot U_{\sigma^{-1}(1)} \ .$$

For m unitaries, observe that the norm of the OI state ranges in $[0, m!]$. We view the norm of the OI state as a fundamental piece of information. As an example, in the case $|\psi\rangle = |0\rangle$, $m = 2$, $U_0 = X$, $U_1 = Z$, the norm is 0:

$$X \cdot Z|0\rangle + Z \cdot X|0\rangle = |1\rangle + (-1) \cdot |1\rangle = 0 \ .$$

On the other hand, for every number of unitaries $m \in \mathbb{N}$ and state $|\psi\rangle$, if $U_1 = U_2 = \cdots = U_m$, one can verify that $m!$ is the norm. The previous work of [Cos22] checks whether (according to the PMF) it is possible to always and unconditionally execute a superposition of unitary orders, for every unitaries $\{U_i\}_{i \in [m]}$ and state $|\psi\rangle$. In our relaxed model, we have two main differences: We take into consideration computational complexity, and we take into consideration the norm of the OI state, in particular as a part of the computational complexity of an algorithm's execution.

Connection to standard quantum computation. Our next observation is that when trying to execute OI transformations in standard quantum mechanics, the consideration of the norm is in fact expressed. More formally, we show there is a quantum algorithm Q that given input unitary circuits $\{U_i\}_{i \in [m]}$ (all act on the same number of qubits n) and an n-qubit state $|\psi\rangle$, executes in time $\text{poly}\left(\sum_{i \in [m]} |U_i|\right)$, where $|U_i|$ is the circuit size of U_i, and outputs the normalized OI state $\mathbf{OI}\left(\{U_i\}_{i \in [m]}, |\psi\rangle\right)$ with success probability $\left(\frac{\|\mathbf{OI}(\{U_i\}_{i \in [m]}, |\psi\rangle)\|}{m!}\right)^2$. We elaborate formally on the above quantum procedure in Section 4.2. There is no known way in quantum mechanics to amplify the probability for obtaining the OI state, other than repeated trials.

Computable Interference of Unitary Orders - Definition. Rather than either the bounded interference in quantum mechanics or an unconditional interference, in this work we investigate the computational power of quantum computing with *computable interference* of unitary orders. In our model, while the probability to succeed in executing order interference is proportional to the norm of the OI state (as per standard quantum mechanics), we allow an investment of computational work (or energy) to amplify the success probability arbitrarily close to 1. The ability to execute computable order interference is formally captured by an object we define, called an Order Interference (OI) oracle $\mathcal{O}_{\mathbf{OI}}$. The full definition (Definition 4.4) of an OI oracle is given in Section 4.1, and for the sake of this Section 1 and Section 2, it will be sufficient to focus on a slightly weaker variant of it (given in Definition 4.2).

The relaxed definition of an OI oracle $\mathcal{O}_{\mathbf{OI}}$ is as follows. Let $\{U_i\}_{i \in [m]}$ a set of $m \in \mathbb{N}$ unitary quantum circuits, all operating on the same number of qubits $n \in \mathbb{N}$, let $|\psi\rangle$ an n-qubit quantum state and let $\lambda \in \mathbb{N}$. An input to $\mathcal{O}_{\mathbf{OI}}$ is a triplet

$$\left(\{U_i\}_{i \in [m]}, |\psi\rangle, \lambda\right) \ ,$$

and takes time complexity $O\left(\lambda + \sum_{i \in [m]} |U_i|\right)$. At the end of execution, with probability

$$\frac{\frac{\|\mathbf{OI}(\{U_i\}_{i \in [m]}, |\psi\rangle)\|}{m!}}{\frac{\|\mathbf{OI}(\{U_i\}_{i \in [m]}, |\psi\rangle)\|}{m!} + \frac{1}{\lambda}} \ , \tag{3}$$

the process measures a "success" signal for the transformation and obtains the normalization of $\mathbf{OI}\left(\{U_i\}_{i \in [m]}, |\psi\rangle\right)$, and with the remaining probability the process measures a "fail" signal and obtains $|0^n\rangle$.

5

1.3 Results

We define a new complexity class named $\mathbf{BQP}^{\text{OI}}$ (Definition 4.5), which stands for $\mathbf{BQP}$ (bounded-error quantum polynomial-time) with access to an Order Interference oracle $\mathcal{O}_{\mathbf{OI}}$. $\mathbf{BQP}^{\text{OI}}$ is defined as the class of problems for which there exists a quantum algorithm with access to an OI oracle, such that the algorithm solves the problem (with bounded error) running in polynomial time in the input size.

   We focus on two of the most well-studied computational tasks in computer science: The Graph Isomorphism problem (GI, Definition 3.9) and the Gap Closest Vector Problem (GapCVP$_{g(n)}$, Definition 3.11). Both GI and GapCVP$_{g(n)}$ for $g(n) := O(n^2)$ are believed by experts to be unsolvable in quantum polynomial time. In particular, GapCVP$_{O(n^2)}$ is a key problem in the theory of computation and have an immense practical significance: Solving GapCVP$_{O(n^2)}$ means solving the Learning With Errors (LWE) problem [Reg09], the computational hardness of which is the central pillar of post-quantum cryptography. We prove the following theorems.

Theorem 1.1. GI $\in \mathbf{BQP}^{OI}$.

Theorem 1.2. *There exists a positive absolute constant $c \in \mathbb{R}_{>0}$ such that* GapCVP$_{c \cdot n^2} \in \mathbf{BQP}^{OI}$.

   We find the precise computational power of $\mathbf{BQP}^{\text{OI}}$ as an interesting open question. While computable OI shows algorithmic results seemingly out of the reach of standard quantum computation, it is not obviously overpowered, in the sense that we did not find a trivial way to solve computational problems that cause major implications to the theory of computational complexity, e.g., problems complete for the classes $\mathbf{NP}$ or $\mathbf{SZK}$.

Scientific Contribution.   On a high-level, this work shows a new connection between information processing models in quantum gravity, and some of the fundamental computational problems in computer science. we make the following contributions to computational complexity and quantum information theory.

- **In computational complexity,** we show a novel parameterization technique for classical circuits $C : \{0,1\}^{k'} \to \{0,1\}^k$, allowing to view invertibility of circuits as a spectrum rather than a binary predicate. We define a new natural and relaxed variant of the known Statistical Difference [SV03] problem (SD, Definition 3.5), called the Sequentially-Invertible Statistical Difference problem (SISD, Definition 4.10). Previous work [GMW91], [GG98] shows how to reduce both GI and GapCVP to SD, and we show new reductions to the relaxed problem SISD. These results are in classical computational complexity theory, independent of quantum computation.

- **In quantum information theory,** we define a new computational model and complexity measure, capturing a natural analogue of quantum computation with unitary order superposition. We observe a connection between the model and the problem of Statistical Difference SD. As a further technical step, we show a new algorithmic technique in the computational model, which in turn shows how for some parameters, the SISD problem can be solved by a quantum computer with unitary order superposition (in formal terms, that for some parameters, SISD $\in \mathbf{BQP}^{OI}$).

Under both disciplines, several open questions stem from our results.

   The remaining of the paper is as follows. In Section 2 we give an overview of the techniques we develop in this work, their connection to new computational problems and also the problems GI and GapCVP. The Preliminaries are given in Section 3. In Section 4 we provide all of the new definitions from this work, including the definition of the OI oracle, the complexity class $\mathbf{BQP}^{\text{OI}}$ and the computational problem SISD. Section 5 contains our main quantum algorithm, and in Sections 6, 7 we provide new reductions for GI and GapCVP, which, together with the results in Section 5, show the containment of the problems in $\mathbf{BQP}^{\text{OI}}$.

2 Technical Overview

In this section we give an overview of the main technical ideas in this work. In Section 2.1 we form an understanding of computational abilities that computable OI has and a standard quantum computer does not seem to posses. We also connect these understandings to the Statistical Difference problem. In Section 2.2 we show a new technique to parameterize invertibility of classical circuits and define a new computational problem, called the Sequentially Invertible Statistical Difference Problem SISD. In Section 2.3 we show a polynomial-time quantum algorithm using an OI oracle, that solves SISD. In Section 2.4 we show classical polynomial-time reductions from the Graph Isomorphism Problem GI and the $O(n^2)$-Gap Closest Vector Problem $\mathsf{GapCVP}_{O(n^2)}$ to the SISD problem.

2.1 Computable Order Interference and Quantum State Generation

Before we are trying to use computable OI to solve new problems, it is important to intuitively understand what it enables that a standard quantum computer does not. In the introduction we saw that a standard quantum computer successfully executes an OI transformation with probability $\left(\frac{\|\mathbf{OI}(\{U_i\}_{i\in[m]},|\psi\rangle)\|}{m!} \right)^2$, that is, in the order of the scaled norm of the OI state. In the setting of computable OI (modeled by an OI oracle $\mathcal{O}_{\mathbf{OI}}$), this probability can be amplified as described in 3. One can verify that to make the success probability 3 at least a constant using computable OI, the complexity of amplification (captured by the parameter λ) has to be at least the inverse of the scaled norm $\Theta\left(\frac{m!}{\|\mathbf{OI}(\{U_i\}_{i\in[m]},|\psi\rangle)\|} \right)$, depends on the wanted constant[4]. However, if we are already allowed to execute in the complexity scales of $\frac{m!}{\|\mathbf{OI}(\{U_i\}_{i\in[m]},|\psi\rangle)\|}$, a standard quantum computer can successfully generate the OI state with high probability: Just by repeatedly trying to execute the OI transformation, where as we know, each attempt succeeds with the bounded probability $\left(\frac{\|\mathbf{OI}(\{U_i\}_{i\in[m]},|\psi\rangle)\|}{m!} \right)^2$. When an OI transformation fails in either of the cases (computable OI or standard quantum computing) the state is considered as destroyed. So, in both cases, with or without computable OI, the computational complexity of successfully executing a single OI transformation is $\approx \frac{m!}{\|\mathbf{OI}(\{U_i\}_{i\in[m]},|\psi\rangle)\|}$.

What is the complexity of executing 2 consecutive OI transformations? For two different sets of unitaries $\{U_{0,i}\}_{i\in[m_0]}$, $\{U_{1,i}\}_{i\in[m_1]}$. Assuming both transformations are successful, this will create a state of different structure than what can be created by a single OI transformation. The gap in complexity between standard quantum computation and computable OI, becomes more apparent when we think of running ℓ consecutive OI transformations, for a general ℓ. Computable OI lets us maintain a probability close to 1 in that case, using only complexity $\approx \ell^3 \cdot \frac{m!}{\|\mathbf{OI}(\{U_i\}_{i\in[m]},|\psi\rangle)\|}$ over all ℓ attempted transformations, while a standard quantum computer needs all ℓ tries to succeed consecutively, which generally has a complexity increasing exponentially with ℓ. One deduction is thus, that the power of computable OI is by executing OI transformations *repeatedly*. Assume we are starting from the state $|0^n\rangle$, and we sequentially execute OI transformations, for different sets of unitaries. The resulting quantum state is

$$\sum_{\sigma_1\in S_{m_1}, \cdots, \sigma_\ell\in S_{m_\ell}} \left(\prod_{i_\ell\in[m_\ell]} U_{\sigma_\ell^{-1}(i_\ell)} \right) \cdots \left(\prod_{i_1\in[m_1]} U_{\sigma_1^{-1}(i_1)} \right) |\psi\rangle , \tag{4}$$

and has a unique structure, corresponding to a uniform superposition over all partial execution orders, where the partial order rule is that for $i,j\in[\ell], i<j$, the execution of unitaries in the i-th set $\{U_{i,k}\}_{k\in[m_i]}$ always come before the execution of unitaries in the j-th set $\{U_{j,k}\}_{k\in[m_j]}$.

---

[4]More generally, to get a success probability of $1-\frac{1}{k}$, executing the OI oracle with $\lambda = k \cdot \frac{m!}{\|\mathbf{OI}(\{U_i\}_{i\in[m]},|\psi\rangle)\|}$ suffices.

The statistical difference problem and quantum state generation. When we think of the ability to generate new quantum states, there comes to mind one of the most fundamental hard problems in computer science – the Statistical Difference (SD) problem [SV03]. The input to the SD problem is a pair of classical circuits (C_0, C_1), and we consider their output distributions (i.e., the output distribution for a uniformly random input string). The problem is to decide the total variation distance (also named statistical distance) between these two distributions. Formally, for functions $a, b : \mathbb{N} \to [0, 1]$ such that $\forall n \in \mathbb{N} : a(n) \leq b(n)$, an input to the $\mathsf{SD}_{a(n), b(n)}$ problem is a pair (C_0, C_1) of classical circuits with the same output size, $C_b : \{0, 1\}^{k_b} \to \{0, 1\}^k$ for some $k_0, k_1, k \in \mathbb{N}$. We denote the problem input size with $n := |C_0| + |C_1|$ and think of the circuits as a pair of sampler circuits, that given uniformly random k_b bits sample a k-bit output. The problem comes with the promise that it is either the case that the statistical distance between the output distributions of the circuits is at most $a(n)$, in which case we denote $(C_0, C_1) \in \prod_{\mathrm{YES}}$, or the statistical distance is more than $b(n)$, in which case we denote $(C_0, C_1) \in \prod_{\mathrm{NO}}$. Generally speaking, the larger the gap $b(n) - a(n)$, the easier the problem. In this work, our focus is on the family of problems $\mathsf{SD}_{a(n), b(n)}$, for functions $a(n), b(n)$ such that there exists a polynomial $\mathrm{poly}(n)$ with $b(n) - 2a(n) + a(n)^2 \geq \frac{1}{\mathrm{poly}(n)}$ [5]. From hereon we drop the subscript $a(n)$, $b(n)$ from $\mathsf{SD}_{a(n), b(n)}$ and write $\mathsf{SD}_{\mathsf{poly}}$ to implicitly refer to the family of problems $\mathsf{SD}_{a(n), b(n)}$ for such functions $a(n), b(n)$. The $\mathsf{SD}_{\mathsf{poly}}$ problem is a notoriously hard problem in computer science (also for quantum computers), and it is at least as hard as some of the widely studied computational problems in cryptography and in combinatorics, some examples include: Quadratic Residuosity, Lattice Isomorphism, Discrete Logarithm, Decisional Diffie-Hellman, Graph Isomorphism, the Gap Closest Vector Problem and the Gap Shortest Vector Problem.

The thing that connects $\mathsf{SD}_{\mathsf{poly}}$ and the ability to generate non-trivial quantum states is a quantum procedure called the Swap Test [BBD+97, BCWDW01]. It is observed in [ATS03] that the $\mathsf{SD}_{\mathsf{poly}}$ problem could be efficiently solved if we had the ability to efficiently generate the output distribution states of classical circuits. If given input (C_0, C_1) we could efficiently generate the normalized k-qubit quantum states,

$$|C_0(R)\rangle := \sum_{x \in \{0,1\}^{k_0}} |C_0(x)\rangle \ ,$$

$$|C_1(R)\rangle := \sum_{x \in \{0,1\}^{k_1}} |C_1(x)\rangle \ ,$$

then we can execute a Swap Test on the pair of states $|C_0(R)\rangle, |C_1(R)\rangle$, which approximates (with some inverse polynomial error) the inner product between the states as vectors. Since the inner product between these states relates to the statistical distance between the output distributions of the circuits C_0, C_1, a Swap Test lets us estimate the statistical distance. Finally, as long as the gap $b(n)^2 - 2a(n) + a(n)^2$ is at least inverse polynomial, the error in the approximation of the Swap Test can be overcome and we can decide whether the pair of circuits is in $\prod_{\mathrm{YES}}$ or $\prod_{\mathrm{NO}}$. So, the $\mathsf{SD}_{\mathsf{poly}}$ problem can be reduced to the quantum state generation of $|C_0(R)\rangle, |C_1(R)\rangle$. With accordance to the hardness of the $\mathsf{SD}_{\mathsf{poly}}$ problem, there is no known way for a quantum computer to generate in polynomial time the output distribution state for a general classical circuit C.

2.2 The Sequentially Invertible Statistical Difference Problem

It is a standard fact in quantum computing that for an efficient classical circuit $C : \{0, 1\}^{k'} \to \{0, 1\}^k$ there is an $(k' + k)$-qubit unitary $U_{x,C}$, of roughly the same computational complexity as C, that maps:

---

[5]The choice of $b(n)^2 - 2a(n) + a(n)^2$ comes from the relation between total variation distance and fidelity, that is, if fidelity between the output distributions of the circuits was the measure determining $a(n)$, $b(n)$, then we will simply be interested in $b(n) - a(n) \geq \frac{1}{\mathrm{poly}(n)}$.

$\forall x \in \{0,1\}^{k'}, y \in \{0,1\}^k : U_{x,C} \cdot |x,y\rangle = |x, y \oplus C(x)\rangle$. It is in particular easy to quantumly generate the $(k'+k)$-qubit state

$$|R, C(R)\rangle := \frac{1}{\sqrt{2^{k'}}} \sum_{x \in \{0,1\}^{k'}} |x, C(x)\rangle \ .$$

The above state is not a valid solution to the quantum state generation problem (and cannot be used to solve SD with Swap Test, as described in Section 2.1). The state we want is the output distribution state $|C(R)\rangle$, where only the output should appear in the superposition, excluding the input. Another known fact in quantum computing is that if a circuit $C : \{0,1\}^k \to \{0,1\}^k$ is both efficiently computable *and* efficiently invertible (i.e., there is a classically efficiently computable $C^{-1} : \{0,1\}^k \to \{0,1\}^k$), then there is an efficient unitary U_C on k (rather than $k'+k$) qubits, mapping $\forall x \in \{0,1\}^k : U_C \cdot |x\rangle = |C(x)\rangle$ (possibly using additional ancillary qubits). In particular, for efficiently computable and invertible circuits, the quantum state generation is quantumly efficiently solvable. However, for all "interesting" cases of the $\mathsf{SD}_{\mathsf{poly}}$ problem (in particular, all instances captured by important cryptographic hard problems), the circuits C_0, C_1 are conjectured to not be efficiently invertible, and are sometimes known to be not invertible at all (i.e., they are not injective). In first glance, it isn't clear how computable OI can help in generating the output distribution states for non-invertible circuits, in a way that a standard quantum computer cannot.

We get a dichotomous situation, where if a circuit $C : \{0,1\}^{k'} \to \{0,1\}^k$ is invertible then we can generate the output distribution state $|C(R)\rangle$ using a standard quantum computer, and if it is not invertible, we don't know a way to generate the state, even with computable OI. Could we look at invertibility in a more high-resolution way? Could we think about invertible classical computation as a numerical parameter, rather than binary?

Sequentially invertible functions. Our next idea is to parameterize invertibility. Our parameterization technique is not necessarily related to quantum computation and may be of independent interest. Note that that sampler circuits are equivalent to randomized circuits, which take the form $C : \{0,1\}^k \times \{0,1\}^{k'} \to \{0,1\}^k$, where the randomness $z \in \{0,1\}^{k'}$ is only considered as auxiliary input, and together with it there's a main input $x \in \{0,1\}^k$. We can thus consider sampler and randomized circuits interchangeably.

Our definition of parameterized invertibility of a function is as follows. Let $f : \{0,1\}^{k'} \to \{0,1\}^k$ a function. For $r \leq k'$, we say that f is r-sequentially invertible if there is an r-invertible function sequence that samples from the output distribution of f [6]. An r-invertible function sequence is a sequence of pairs of functions $(f_{i,\to}, f_{i,\leftarrow})_{i \in [\ell]}$ such that,

- There exists $\bar{k} \in \mathbb{N}$ such that all functions have input and output size $\bar{k}$, and randomness bounded by r. Additionally, each pair of opposite-direction functions use the same amount of randomness. Formally, for every $i \in [\ell]$ there exists $r_i \leq r$ such that

$$\forall Y \in \{\to, \leftarrow\} : f_{i,Y} : \{0,1\}^{\bar{k}} \times \{0,1\}^{r_i} \to \{0,1\}^{\bar{k}} \ .$$

- The function directions are inverses of each other, per hard-wired randomness, that is:

$$\forall i \in [\ell], z \in \{0,1\}^{r_i}, x \in \{0,1\}^{\bar{k}} : x = f_{i,\leftarrow}(f_{i,\to}(x;z);z) \ .$$

We say that the sequence samples from f, if $\bar{k} = k$, and we have the following equivalence of distributions,

$$\left\{ f(z) \mid z \leftarrow \{0,1\}^{k'} \right\} \equiv$$

---

[6]The full definition of $(r, t, \ell, \varepsilon)$-sequential invertibility, which takes into consideration more parameters and not only r, is given in Section 4.3. For the needs for this overview, we give the minimal definitions.

$$\left\{ f_{\ell,\to}\left(\cdots f_{2,\to}\left(f_{1,\to}\left(0^{\bar{k}};z_1\right);z_2\right)\cdots;z_\ell\right) \;\middle|\; z_1 \leftarrow \{0,1\}^{r_1},\cdots,z_\ell \leftarrow \{0,1\}^{r_\ell} \right\} \; .$$

The intuition is that the smaller the parameter r is, "the more the function is invertible". We give a few examples. For any invertible function $f : \{0,1\}^k \to \{0,1\}^k$, f is 0-sequentially invertible (for $\ell = 1$, that is, by a single pair $(f_{1,\to}, f_{1,\leftarrow})$). $f_\oplus : \{0,1\}^{2\cdot k} \to \{0,1\}^k$, that for two strings $a, b \in \{0,1\}^k$ outputs their bit-wise logical XOR: $f_\oplus(a, b) := a \oplus b$, is 1-sequentially invertible (by a sequence of $\ell = k$ pairs), and $f_\wedge : \{0,1\}^{2\cdot k} \to \{0,1\}^k$, that for two strings $a, b \in \{0,1\}^k$ outputs their bit-wise logical AND: $f_\wedge(a, b) := a \wedge b$, is 2-sequentially invertible (again, by a sequence of k pairs). We leave it as an instructive exercise for the reader to verify these claims.

Definition of the SISD **problem.** We combine the ideas of sequential invertibility with the SD problem, and define the $(a(n), b(n), r(n))$-Sequentially Invertible Statistical Difference Problem $\left(\mathsf{SISD}_{a(n),b(n),r(n)}\right)$ for functions $a(n), b(n) : \mathbb{N} \to [0,1]$, $r(n) : \mathbb{N} \to \mathbb{N}$, such that for every $n \in \mathbb{N}$, $a(n) \leq b(n)$ and $r(n) \in \{0, 1, \cdots, n\}$. An input is a pair of sequences,

$$\left(\left(C^0_{i,\to}, C^0_{i,\leftarrow}\right)_{i\in[\ell]}, \left(C^1_{i,\to}, C^1_{i,\leftarrow}\right)_{i\in[\ell]} \right) \;,$$

where for each $b \in \{0,1\}$, $\left(C^b_{i,\to}, C^b_{i,\leftarrow}\right)_{i\in[\ell]}$ is an $r(n)$-invertible circuit sequence, where n is the input size,

$$n := \left| \left(\left(C^0_{i,\to}, C^0_{i,\leftarrow}\right)_{i\in[\ell]}, \left(C^1_{i,\to}, C^1_{i,\leftarrow}\right)_{i\in[\ell]} \right) \right| \;,$$

and such that both sequences have the same input and output size k, that is, for every $i \in [\ell]$, $b \in \{0,1\}$, $Y \in \{\to, \leftarrow\}$,

$$C^b_{i,Y} : \{0,1\}^k \times \{0,1\}^{r^b_i(n)} \to \{0,1\}^k \;.$$

Similarly to the case of the regular Statistical Distance problem $\mathsf{SD}_{a(n),b(n)}$, we have the following promise on the gap of the statistical distance between the circuits. Formally, for $b \in \{0,1\}$ denote

$$D_b := \left\{ C^b_{\ell,\to}\left(\cdots C^b_{2,\to}\left(C^b_{1,\to}\left(0^{k(n)};z_1\right);z_2\right)\cdots;z_\ell\right) \right\} \;,$$

where the randomness sequence is sampled in both cases uniformly $z_1 \leftarrow \{0,1\}^{r^b_1(n)}, \cdots, z_\ell \leftarrow \{0,1\}^{r^b_\ell(n)}$ for the suitable $b \in \{0,1\}$. We denote the statistical distance between the distributions D_0, D_1 with d, and it is promised that either $d \leq a(n)$, in which case we say that the input belongs to $\prod_{\mathrm{YES}}$, or we have $d > b(n)$, in which case we determine that the input belongs to $\prod_{\mathrm{NO}}$. The goal of the problem is to decide whether the input belongs to $\prod_{\mathrm{YES}}$ or to $\prod_{\mathrm{NO}}$. As our previous convention for the SD problem, we also drop the subscript $a(n), b(n)$ to implicitly be any pair of functions with difference $b(n)^2 - 2a(n) + a(n)^2$ lower-bounded inverse polynomially, and write $\mathsf{SISD}_{\mathsf{poly},\,r(n)}$.

2.3 How to Solve $\mathsf{SISD}_{\mathsf{poly},\,O(\log(n))}$ with Computable Order Interference

The $\mathsf{SISD}_{\mathsf{poly},\,r(n)}$ problem is a private case of the $\mathsf{SD}_{\mathsf{poly}}$ problem (for any randomness $r(n) \in [0, n]$), so a statement that holds for general circuits in $\mathsf{SD}_{\mathsf{poly}}$ also holds for $\mathsf{SISD}_{\mathsf{poly},\,r(n)}$. In order to solve $\mathsf{SISD}_{\mathsf{poly},\,O(\log(n))}$ we will show how to efficiently solve the quantum state generation problem for sequentially invertible circuits with randomness $O(\log(n))$. More precisely, given an input $r(n)$-invertible circuit sequence $C = (C_{i,\to}, C_{i,\leftarrow})_{i\in[\ell]}$, such that $r(n) = O(\log(n))$, we show how to generate the output distribution state

$$|C(R)\rangle := \sum_{\substack{z_1\in\{0,1\}^{r_1}, \\ \vdots \\ z_\ell\in\{0,1\}^{r_\ell}}} \left|C_{\ell,\to}\left(\cdots C_{2,\to}\left(C_{1,\to}\left(0^k;z_1\right);z_2\right)\cdots;z_\ell\right)\right\rangle \;, \tag{5}$$

by a quantum computer with access to a computable order interference oracle $\mathcal{O}_{\mathbf{OI}}$, in polynomial time.

We can think of the superposition in the state $|C(R)\rangle$ as a tree – in level $i \in \{0, 1, \cdots, \ell\}$ we have a superposition over the randomness of all i circuits $C_{1,\rightarrow}, \cdots, C_{i,\rightarrow}$, and the state $|C(R)\rangle$ is simply the full tree of depth ℓ. We can thus attempt to start from level 0, i.e., the state $|0^k\rangle$, and progress one level at a time. Assume we have a superposition over level $i \in \{0, 1, \cdots, \ell-1\}$, that is, we have the state,

$$|C_{1,\cdots,i}(R_{1,\cdots,i})\rangle := \sum_{\substack{z_1 \in \{0,1\}^{r_1}, \\ \vdots \\ z_i \in \{0,1\}^{r_i}}} \left|C_{i,\rightarrow}\left(\cdots C_{1,\rightarrow}\left(0^k; z_1\right)\cdots; z_i\right)\right\rangle , \qquad (6)$$

and we want to obtain the state of the next level of the distribution (with high probability),

$$|C_{1,\cdots,i+1}(R_{1,\cdots,i+1})\rangle := \sum_{\substack{z_1 \in \{0,1\}^{r_1}, \\ \vdots \\ z_{i+1} \in \{0,1\}^{r_{i+1}}}} \left|C_{i+1,\rightarrow}\left(\cdots C_{1,\rightarrow}\left(0^k; z_1\right)\cdots; z_{i+1}\right)\right\rangle . \qquad (7)$$

For each randomness $z_{i+1} \in \{0,1\}^{r_{i+1}}$, the circuit $C_{i+1,\rightarrow}(\cdot; z_{i+1})$ is efficiently computable and invertible, which, as we know, means we have the efficient k-qubit unitary $U_{i+1,z_{i+1}}$, mapping for every $x \in \{0,1\}^k$, $U_{i+1,z_{i+1}} \cdot |x\rangle = |C_{i+1,\rightarrow}(x; z_{i+1})\rangle$. Given this structure, observe that the transformation we want to apply to the state $|C_{1,\cdots,i}(R_{1,\cdots,i})\rangle$ is,

$$\sum_{z_{i+1} \in \{0,1\}^{r_{i+1}}} U_{i+1,z_{i+1}} ,$$

which will exactly turn the state into $|C_{1,\cdots,i+1}(R_{1,\cdots,i+1})\rangle$. An additional way to look at this is that we want to apply a superposition over unitary *choices*, rather than orders, over the set $\{U_j\}_{j \in [m]}$, where we changed notation by $m = 2^{(r_{i+1})}$, $\forall j \in [m] : U_j := U_{i+1,j}$. The amount of randomness $r_{i+1} \leq r(n)$ is logarithmic in n, thus the number of unitaries in the set $\{U_j\}_{j \in [m]}$ is polynomial in n (and all of these unitaries are of polynomial time complexity). In order to finish our algorithm for state generation, it remains to only explain how to efficiently execute the above transformation of superposition of choice, with high probability.

Our problem is reduced to a clean setting at this point. Given input a set of m unitaries $\{U_i\}_{i \in [m]}$ and a state $|\psi\rangle$, we want to execute an interference of choices rather than orders, that is, to obtain the normalization of

$$\sum_{i \in [m]} U_i \cdot |\psi\rangle ,$$

with high probability. We wish to use OI transformation for this task. As in the case of OI transformations, we want the (pre-amplification) success probability of the transformation to be proportional to the norm of the generated state divided by the number of terms in the sum. In our case,

$$\frac{\left\| \sum_{i \in [m]} U_i \cdot |\psi\rangle \right\|}{m} .$$

Finally, before proceeding to solution, observe that being able to execute choice interference transformations with the above success probability solves the quantum state generation problem for $O(\log)$-sequentially invertible circuits, and thus overall $\mathsf{SISD}_{\mathsf{poly},O(\log(n))} \in \mathbf{BQP^{OI}}$.

Order interference plus small quantum memory implies choice interference. Generally when we execute OI transformation, the significant part of computational complexity is $m!$ divided by the norm of

the OI state. As we explained earlier, the norm of the OI state ranges in $[0, m!]$, and the observation that leads our solution is the fact that there is a way to control the norm, by making unitary paths converge (while paying with limiting the variety of the superposition). For example, by taking all unitaries to be identical $U_1 = \cdots = U_m$ the norm is $m!$, and the success probability is 1. However, this is equivalent to simply executing U_1^m, which can be done by standard quantum computer.

Let $n \in \mathbb{N}$ the number of qubits in the state $|\psi\rangle$ (and the number of qubits that each of the unitaries $\{U_i\}_{i \in [m]}$ act on), and consider adding an ancilla of $\lceil \log m \rceil$ qubits, so the state we are operating on is $|\psi\rangle|0^{\lceil \log m \rceil}\rangle$. For each unitary U_i we make a slight change, and execute the modified unitary $\tilde{U}_i$ which consists of two steps. (1) Execute the circuit U_i^c, which executes U_i on the left n qubits, conditioned on the state of the $\lceil \log m \rceil$ right qubits being $|0^{\lceil \log m \rceil}\rangle$. (2) Execute the unitary U_{+1} on the right $\lceil \log m \rceil$ qubits, which treats the qubits as a binary representation of a number in $\{0, 1, \cdots, m-1, \cdots, 2^{\lceil \log m \rceil} - 1\}$, and increments by 1 modulo $2^{\lceil \log m \rceil}$. Here, we use the following fact, which we will also use later:

Fact 2.1. *For every numbers $a, N \in \mathbb{N}$ such that $a \leq N$, there is a unitary circuit $U_{a,N}$ on $\lceil \log(N) \rceil$ qubits, executing in time $\mathsf{poly}\left(\log\left(a \cdot N\right)\right)$ that maps*

$$\forall b \in \{0, 1, \cdots, N-2, N-1\} : U_{a,N} \cdot |b\rangle = |b + a \ (mod) \ N\rangle \ .$$

That is, if we know the numbers a, N not as part of the computation but in advance, then it is possible to unitarily and efficiently add a modulo N. One can verify that the above fact follows from the function that adds a modulo N is both efficiently computable and invertible.

We consider what happens to the modified state $|\psi\rangle|0^{\lceil \log m \rceil}\rangle$, in each of the unitary execution orders $\sigma \in S_m$, for the modified unitaries $\tilde{U}_i$. Two things can be verified by the reader.

- The first thing to see is that for $i \in [m]$ and any permutation $\sigma \in S_m$ such that $\sigma^{-1}(i) = 1$, the state in the left n qubits at the end of the computation is $U_i \cdot |\psi\rangle$. In simple terms, the only unitary that effectively executes on the side of the input state $|\psi\rangle$ is the first one in the ordering.

- The second thing to notice is that when we look at the state of the right $\lceil \log(m) \rceil$-qubit counter register at the end of the computation, for any permutation $\sigma \in S_m$, the state is always $m \mod 2^{\lceil \log(m) \rceil}$.

Together, this means that the OI state is,

$$\mathbf{OI}\left(\{\tilde{U}_i\}_{i \in [m]}, |\psi\rangle|0^{\lceil \log(m) \rceil}\rangle\right) := \sum_{\sigma \in S_m} \left(\prod_{i \in [m]} \tilde{U}_{\sigma^{-1}(i)}\right) \cdot |\psi\rangle|0^{\lceil \log(m) \rceil}\rangle$$

$$= \sum_{i \in [m]} \left(\sum_{\sigma \in S_{m-1}}\right) (U_i \cdot |\psi\rangle) \otimes |m \mod 2^{\lceil \log(m) \rceil}\rangle$$

$$= (m-1)! \sum_{i \in [m]} (U_i \cdot |\psi\rangle) \otimes |m \mod 2^{\lceil \log(m) \rceil}\rangle \ .$$

So, if we get the normalized OI state, we can trace out the right $\lceil \log(m) \rceil$-qubit register without damaging the quantum information in the left n qubits, and get the choice interference state. Finally, the scaled

norm of the OI state is thus

$$
\frac{\left\|\mathbf{OI}\left(\{\tilde{U}_i\}_{i\in[m]}, |\psi\rangle|0^{\lceil\log(m)\rceil}\rangle\right)\right\|}{m!}
$$
$$
= \frac{\left\|(m-1)!\sum_{i\in[m]}(U_i\cdot|\psi\rangle)\otimes|m\bmod 2^{\lceil\log(m)\rceil}\rangle\right\|}{m!}
$$
$$
= \frac{\left\|\sum_{i\in[m]}(U_i\cdot|\psi\rangle)\otimes|m\bmod 2^{\lceil\log(m)\rceil}\rangle\right\|}{m}
$$
$$
= \frac{\left\|\sum_{i\in[m]}U_i\cdot|\psi\rangle\right\|}{m} \ ,
$$

which concludes the proof.

2.4 Reducing GI and GapCVP to $\mathsf{SISD}_{\mathsf{poly},O(\log(n))}$

In the last part of the Technical Overview we will show that two problems that are believed to be hard for quantum computers, are solvable by a quantum computer with computable order interference, by reducing to the $\mathsf{SISD}_{\mathsf{poly},O(\log(n))}$ problem. In the first part of this section we show the reduction from GI which is more immediate, and in the second part we show the more involved reduction from $\mathsf{GapCVP}_{O(n^2)}$. To this end, we will first review the existing reduction from each of the problems to the more general $\mathsf{SD}_{\mathsf{poly}}$ problem (which we don't know how to solve), and then move to our new reductions, to the easier problem of $\mathsf{SISD}_{\mathsf{poly},O(\log(n))}$.

The Graph Isomorphism Problem and Statistical Difference. An input to the GI problem is a pair of (simple, undirected) graphs (G_0, G_1), and the problem is to decide whether the pair of graphs are isomorphic or not. More formally, for $n \in \mathbb{N}$ we define a graph as $G = ([n], E)$, where $E \subseteq [n]\times[n]$ is the set of edges. Two graphs are isomorphic iff there exists an isomorphism between them, that is, a bijection (or, permutation) f on $[n]$ such that for every pair $u, v \in [n]$: $\{u,v\}\in E_0 \iff \{f(u),f(v)\}\in E_1$, where for $b\in\{0,1\}$, E_b is the set of edges in the graph G_b.

As stated earlier in Section 2.1, there is an existing reduction from GI to the standard statistical difference problem $\mathsf{SD}_{\mathsf{poly}}$. More precisely, [GMW91] shows a statistical zero-knowledge protocol for the GI problem, out of which there can be derived the reduction $\mathsf{GI} \leq_p \mathsf{SD}_{0,1}$. The reduction to $\mathsf{SD}_{0,1}$ means that for an input pair (G_0, G_1) to the GI problem, if the graphs are isomorphic then the output pair of circuits (C_0, C_1) are such that the statistical distance between their output distributions is 0, and if the graphs are not isomorphic, the distance is 1, which means that the supports of the output distributions are disjoint.

The reduction is as follows. Given the input graphs (G_0, G_1), the first circuit C_0 is defined such that for input randomness r of polynomial length, it samples a uniformly random permutation $\sigma \in S_n$ on the elements $[n]$, applies σ to the graph G_0, and the output of C_0 is the σ-permuted graph $\sigma(G_0)$[7]. The circuit C_1 is identical, only that it applies σ to the second graph, G_1. To see the correctness of the reduction, recall that for two isomorphic graphs, their sets of all isomorphic graphs are equal, thus in that case, the circuits C_0, C_1 sample from the exact same distribution (uniform distribution over their isomorphic graphs!) and the statistical distance is 0. In the other case where the graphs are not isomorphic, there does not exist a permutation σ that will map one graph to the other, and in particular, the sets of permutations of the two graphs will never intersect, which will result in a statistical distance of 1.

The reduction $\mathsf{GI} \leq_p \mathsf{SISD}_{(0,1,\log(n))}$. It remains to give a more fine-grained reduction for GI. The goal of our reduction is to stick to the previous reduction of [GMW91], in the sense that the circuit C_b will

---

[7]Recall that permuting G means applying σ to the vertices of G, but keeping the edges connecting the same vertex numbers.

still aim to output a random permutation of the graph G_b (for $b \in \{0, 1\}$), but additionally, the sampling will be done in a $\log(n)$-sequentially invertible manner.

Our reduction follows by two arguments. The first step is that constant permutation mappings are efficiently computable and invertible. Formally, assume that we use adjacency matrix representation of a graph, so n^2 bits are sufficient to represent any n-vertex graph, then for a fixed permutation $\sigma \in S_n$, the circuit $C_\sigma : \{0, 1\}^{n^2} \to \{0, 1\}^{n^2}$ that permutes a given input graph G with the permutation σ, is both efficiently computable and efficiently invertible. The second step is that a random permutation over $[n]$, which has $n!$ options, can be sampled in many small steps, each having a small number of options. Specifically, the Fisher-Yates shuffle [FY53], shows how to sample a random permutation in $n - 1$ steps, each step $i = n, n - 1, \cdots, 3, 2$ having i options, and importantly, each step can be thought of as a fixed permutation by itself, which means the circuit that implements it is efficiently computable and invertible. Formally, for a decreasing $i = n, n - 1, \cdots, 3, 2$, pick a random number $j \in [i]$, and swap between i and j.

It remains to compile our observations into a $\log(n)$-invertible circuit sequence. Consider $n - 1$ circuit pairs $(C_{i,\rightarrow}, C_{i,\leftarrow})_{i \in \{n, n-1, \cdots, 3, 2\}}$, such that $C_{i,\rightarrow} : \{0, 1\}^{n^2} \times \{0, 1\}^{\lceil \log(n) \rceil} \to \{0, 1\}^{n^2}$ maps between n-vertex graphs and uses the bits of randomness to pick a random element in $j \in [i]$, and then applies a swap between vertices i and j. Per fixed randomness $j \in [i]$, the circuit $C_{i,\rightarrow}(\cdot, j)$ performs a swap between vertices i, j, which one can observe to be a fixed permutation.

There are some technicalities to deal with (like that sometimes ranges of randomness are represented by binary strings but are not of a size which is a power of 2, which we use rejection sampling to deal with) in the full proof, but overall, the above circuit sequence implements the Fisher Yates shuffle. By generating one circuit sequence $\left(C_{i,\rightarrow}^0, C_{i,\leftarrow}^0\right)_{i \in \{n, n-1, \cdots, 3, 2\}}$ to permute the graph G_0 and another $\left(C_{i,\rightarrow}^1, C_{i,\leftarrow}^1\right)_{i \in \{n, n-1, \cdots, 3, 2\}}$ to permute G_1, both according to the Fisher Yates algorithm, which we break into small swap circuits, the correctness of the reduction follows similar lines to that of [GMW91]. The full proof is given in Section 6.

The $g(n)$-Gap Closest Vector Problem. For a gap function $g : \mathbb{N} \to \mathbb{R}_{\geq 1}$, an input to the $\mathsf{GapCVP}_{g(n)}$ problem is a triplet $(\mathbf{B}, \mathbf{t}, d)$, such that

- $\mathbf{B} = \{\mathbf{B}_1, \mathbf{B}_2, \cdots, \mathbf{B}_n\}$ is a basis for $\mathbb{R}^n$ with integer vectors, i.e., $\mathbf{B} \subseteq \mathbb{Z}^n$.

- $\mathbf{t} \in \mathbb{Z}^n$ is an arbitrary integer vector.

- $d \in \mathbb{R}_{\geq 0}$ is a non-negative real number.

We define $\mathcal{L}_{\mathbf{B}}$, the lattice given by the basis $\mathbf{B}$, as the set of all integer linear combinations of the vectors in $\mathbf{B}$, that is:

$$\mathcal{L}_{\mathbf{B}} := \left\{ \mathbf{a} \in \mathbb{Z}^n \mid \exists \mathbf{s} \in \mathbb{Z}^n : \mathbf{a} = \mathbf{B} \cdot \mathbf{s} \right\} .$$

The instance comes with a promise: It is either the case that the vector $\mathbf{t}$ is d-close to the lattice i.e., $\mathrm{dist}\,(\mathcal{L}_{\mathbf{B}}, \mathbf{t}) \leq d$, in which case we denote $(\mathbf{B}, \mathbf{t}, d) \in \prod_{\mathrm{YES}}$, or the vector is at least $(d \cdot g(n))$-far from the lattice i.e., $\mathrm{dist}\,(\mathcal{L}_{\mathbf{B}}, \mathbf{t}) > d \cdot g(n)$, in which case we denote $(\mathbf{B}, \mathbf{t}, d) \in \prod_{\mathrm{NO}}$. In simple words, a solving algorithm for $\mathsf{GapCVP}_{g(n)}$ gets the promise that the vector $\mathbf{t}$ is either close or far away from the lattice $\mathcal{L}_{\mathbf{B}}$, and needs to decide which is the case. The bigger the gap $g(n)$ is, the easier the problem becomes[8]. What we will show is that for $g(n) = O\left(n^2\right)$ there is a reduction $\mathsf{GapCVP}_{g(n)} \leq_p \mathsf{SISD}_{\mathsf{poly}, O(\log(n))}$. The inspiration and starting point of our reduction is the reduction $\mathsf{GapCVP}_{\sqrt{\frac{n}{O(\log(n))}}} \leq_p \mathsf{SD}_{\mathsf{poly}}$ by Goldreich and Goldwasser [GG98], which we recall next.

---

[8]In particular, it is known that for an exponential $g(n) \approx 2^n$ the GapCVP_{2^n} problem is solvable in classical polynomial time.

The Goldreich-Goldwasser reduction $\mathsf{GapCVP}_{\sqrt{\frac{n}{O(\log(n))}}} \leq_p \mathsf{SD}_{\left(1 - \frac{1}{n^c}, 1\right)}$. The [GG98] reduction shows how, for a smaller gap $\sqrt{\frac{n}{O(\log(n))}}$ (that is, a harder version of GapCVP), we can reduce the Gap Closest Vector Problem to the Statistical Difference Problem SD. Given a $\mathsf{GapCVP}_{g(n)}$ instance $(\mathbf{B}, \mathbf{t}, d)$ (for $g(n) = \sqrt{\frac{n}{O(\log(n))}}$), the reduction executes in classical polynomial time and outputs (C_0, C_1), such that the mapping should preserve $\prod_{\mathrm{YES}}$ and $\prod_{\mathrm{NO}}$ possession. Given the input, the [GG98] reduction first defines an upper bound $M := M_{(\mathbf{B}, \mathbf{t})} := 2^n \cdot \max_{\mathbf{v} \in \{\mathbf{b}_1, \cdots, \mathbf{b}_n, \mathbf{t}\}} \|\mathbf{v}\|$. The upper bound is used to treat an inherently infinite lattice in a finite way, by considering only M-bounded coordinates. The definition of the circuits C_0, C_1 is as follows. The circuit C_0 first samples a uniformly random M-bounded coordinates vector $\mathbf{s} \in \mathbb{Z}_M^n$ and transforms the coordinates to a lattice vector $\mathbf{v} := \mathbf{B} \cdot \mathbf{s}$. The final output of C_0 is a uniformly random point $\mathbf{p}$ inside the n-dimensional ball of radius $\frac{d \cdot g(n)}{2}$ around $\mathbf{v}$. The circuit C_1 is the same, with one change: At the end, we add $\mathbf{t}$ to $\mathbf{p}$. Note that if the distance of the vector $\mathbf{t}$ from the lattice is d, then it can be written as $\mathbf{t} = \mathbf{a} + \mathbf{e}$, such that $\mathbf{a} \in \mathcal{L}_{\mathbf{B}}$ and $\|\mathbf{e}\| = d$, such that $\mathbf{e}$ is the shortest vector which satisfies this equality.

The rationale behind the reduction is that when having a uniform distribution over a lattice, then adding a lattice vector to the output keeps the distribution identical. This logic stays consistent for a distribution that samples balls around the lattice points. So, when looking at the change between C_0, C_1, the lattice component $\mathbf{a}$ in the vector $\mathbf{t}$ gets "swallowed", and the only vector that skews the distribution is $\mathbf{e}$. One can imagine how adding $\mathbf{e}$ pushes all balls around the lattice points in its direction and length, and this is exactly the move from the output distribution of C_0 to the output distribution of C_1. If $(\mathbf{B}, \mathbf{t}, d) \in \prod_{\mathrm{YES}}$ then the skewing vector $\mathbf{e}$ has short length $\leq d$ compared to the radius $\frac{d \cdot g(n)}{2}$ of the balls, which makes the balls have significant intersection, which in turn implies an upper bounded statistical distance between the two circuits (specifically, the statistical distance is $\leq 1 - \frac{1}{n^c}$ for some constant $c > 0$) and thus $(C_0, C_1) \in \prod_{\mathrm{YES}}$. If $(\mathbf{B}, \mathbf{t}, d) \in \prod_{\mathrm{NO}}$ and $\mathbf{e}$ has length $> d \cdot g(n)$, one can observe how it pushes the n-balls (which have radius $\frac{d \cdot g(n)}{2}$, which is in turn half of the distance to the vector $\mathbf{e}$) to be completely disjoint from the original ones, which implies a statistical distance of 1 and thus $(C_0, C_1) \in \prod_{\mathrm{NO}}$.

Our reduction $\mathsf{GapCVP}_{O(n^2)} \leq_p \mathsf{SISD}_{\left(\frac{1}{2} + \frac{1}{2^n}, 1, 1\right)}$. The [GG98] reduction maps to circuits (C_0, C_1) that sample a uniform point $\mathbf{p}$ inside the radius-$\frac{d \cdot g(n)}{2}$, n-dimensional ball, around a uniform lattice vector $\mathbf{v}$. While sampling a random lattice vector can be done in a sequentially-invertible manner, sampling a uniform point inside the n-ball by a sequentially invertible algorithm seems like a fundamental problem. More specifically (but still in a nutshell), the reason that sampling from the n-ball seems hard, is that known algorithmic techniques to get a uniformly random sample from the n-ball include *normalization*. That is, we sample from some large set S, vectors that are not necessarily inside the wanted radius of the n-ball, and then normalize. The problem arises because normalization is a function of the entire vector, and it is inherently a non-invertible function (for example, all vectors of the same direction but different lengths, will map to the same normalized vector).

In our reduction we will still aim to sample from a distribution P around uniform lattice vectors, but not from the uniform n-ball. We note that our distribution P should have two main characteristics.

- The distribution P should not have a preferred direction, out of the n directions in $\mathbb{Z}^n$. This requirement is to deal with arbitrary locations of the vector $\mathbf{t}$ with respect to the lattice $\mathcal{L}_{\mathbf{B}}$ (that is, if P that the reduction picks prefers some direction $i \in [n]$, then the vector $\mathbf{t}$ can be chosen as a function of that direction, tricking the reduction).

- We imagine the support of P as an n-dimensional shape. The distribution P should be as dense as possible in its shape. The density requirement emerges from the case where $(\mathbf{B}, \mathbf{t}, d) \in \prod_{\mathrm{YES}}$ – in that case we sample uniform samples from the shape around the lattice vectors, and the more

the shape is sparse, the less our ability to give a good upper bound on the total variation distance between the output distributions of the circuits C_0, C_1.

It turns out that the distribution which maximizes the above two requirements is indeed the uniform distribution inside the n-ball. The challenge in our reduction is to find a proper distribution P to sample from, and then design a $O(\log(n))$-sequentially-invertible sampling algorithm. In fact, we will show a distribution P with a 1-sequentially-invertible sampling algorithm.

The distribution P. The distribution P we choose is the n-dimensional discrete truncated Gaussian distribution D_B^n (as per Definition 3.12, $B \in \mathbb{N}$ is the truncation parameter, assume it to be $\approx \frac{d \cdot g(n)}{2 \cdot \sqrt{n}}$). It is a known fact that D_B^n is invariant under rotations in n-space, which exactly means it does not prefer any direction. In terms of sparsity, the Gaussian distribution is obviously less sparse than the uniform n-ball, which is (part of) the reason that we need a bigger gap $g(n)$ for which we know how to reduce $\mathsf{GapCVP}_{g(n)}$ to SISD, compared to the better gap $\sqrt{\frac{n}{O(\log(n))}}$ from the previous reduction. As part of the full proof of the reduction (given in Section 7) we analyze the Gaussian distribution's sparsity in this manner, and the loss of hardness (which translates to the larger $g(n) \approx n^2 >> \sqrt{\frac{n}{O(\log(n))}}$) that the sparsity incurs. It remains to show a sequentially-invertible sampling algorithm for $P := D_B^n$. One of the basic properties of the Gaussian distribution says that sampling from D_B^n can be done by sampling n i.i.d. samples from D_B (one sample for each of the coordinates). All we need to find is a sequentially-invertible algorithm to sample from the discrete Gaussian distribution D_B.

Sampling from D_B has a plethora of existing techniques and algorithms (e.g., [GPV08, Pei10, MP12, Ros19]), but it is unclear how to show that these can be executed by a sequentially invertible algorithm. In fact, we did not find any sequentially invertible algorithm that samples from a distribution with negligible total variation distance from D_B. We give the intuitive steps behind our solution in the following.

- Recall one of the basic theorems of probability theory - The Central Limit Theorem (CLT). The CLT says that when we take the average of a collection of κ i.i.d. random numerical variables $X_1, \cdots, X_\kappa$, this average $\frac{\sum_{i \in [\kappa]} X_i}{\kappa}$ approaches the Gaussian distribution as κ approaches infinity. This helps us because now, instead of sampling from the Gaussian distribution (which we do not know how to do with a sequentially invertible algorithm), we can repeatedly sample from any distribution. While the CLT asserts closeness to the Gaussian distribution for *some* number of summands, it does not guarantee a rate of convergence, as the rate depends on the chosen distribution of the summands $\{X_i\}_{i \in [\kappa]}$.

- We observe that the uniform distribution has two properties that work in our favour, for the summands $\{X_i\}_{i \in [\kappa]}$. Using a strengthening of the CLT (a Berry-Esseen-type Theorem), it can be shown that the uniform distribution gives a good convergence rate to the Gaussian distribution, even in total variation distance and even in the discrete case. Formally, in Section 7 we show that summing κ i.i.d. samples of the uniform distribution over $-2^\beta, \cdots, -1, 0, 1, \cdots, 2^\beta$ for $2^\beta \approx \frac{B}{\kappa}$ has total variation distance bounded by $O\left(\frac{1}{\sqrt{\kappa}}\right)$ to the discrete Gaussian distribution D_B.

- A second property of the uniform distribution is that it can be sampled by a sequentially-invertible algorithm, in fact, by a 1-invertible circuit sequence. To see this, observe that to sample uniformly at random from $\{0, 1, \cdots, 2^{\beta+1} - 1\}$, all we need to do is to break the sampled number into its summation as powers of 2, and in the i-th circuit for $i \in [\beta]$, we add (or not) the number 2^i. Summing random powers of 2 up to power β exactly gives a uniformly random distribution in $\{0, 1, \cdots, 2^{\beta+1} - 1\}$. At the end we can deterministically move the distribution to center around zero, by subtracting 2^β.

- To see why the sequence is sequentially invertible, observe that for every $i \in [\beta]$, for every possible randomness for the circuit C_i (which is just one bit $b_i \in \{0, 1\}$), the fixed-randomness

16

circuit $C_i(\cdot; b_i)$ always adds modulo $2^{\beta+1}$ the same number $b_i \cdot 2^i$. By recalling Fact 2.1 once again,

To conclude, in our reduction, the circuit C^0 samples a uniformly random lattice vector $\mathbf{v}$, and then adds an approximate discrete Gaussian vector $\mathbf{e}$, where the approximation is done by summing i.i.d. samples from the uniform distribution. The circuit C^1 is the same, only that we add $\mathbf{t}$ in the end. Both circuits are shown to be 1-sequentially invertible. We refer the reader to Section 7 for the full details of the proof, including the analysis of the total variation distance between the circuits.

3 Preliminaries

In this section we introduce previously existing notions and definitions from the literature that are relevant to this work.
We use the following notions and notations, most of which are standard in quantum computation and complexity.

- For $n \in \mathbb{N}$, define $[n] := \{1, 2, 3, \cdots, n\}$.

- In this work, when we use the notation $\log(\cdot)$ we implicitly refer to the base-2 logarithm function $\log_2(\cdot)$.

- In this work, the only non-standard notation will be for iterated matrix multiplication. Specifically, our default convention for iterated matrix multiplication, using the notation $\prod$, will be by enumerating from right to left (unlike the usual, left to right). Specifically, for a natural number $m \in \mathbb{N}$ and square complex matrices $\{M_i\}_{i \in [m]}$ ($\forall i \in [m] : M_i \in \mathbb{C}^{n \times n}$ for some $n \in \mathbb{N}$), we define:

$$\prod_{i \in [m]} M_i := M_m \cdot M_{m-1} \cdots M_2 \cdot M_1 \ .$$

- We follow the standard Dirac notations for quantum states in quantum information processing. For a variable x (i.e., notation) that denotes a classical binary string, i.e., $x \in \{0, 1\}^*$, the notation $|x\rangle$ refers to the pure quantum state of $|x|$ qubits, which equals that standard basis element of the space $\mathbb{C}^{2^{|x|}}$ that corresponds to the classical string x. That is,

$$\forall n \in \mathbb{N} : \forall x \in \{0, 1\}^n : |x\rangle := e_x := (0_1, 0_2, \cdots, 0_{x-1}, 1_x, 0_{x+1}, \cdots, 0_{2^n-1}, 0_{2^n})^T \ .$$

In case $x \in \mathbb{Z}$ is a natural number, the notation $|x\rangle$ first translates x to its $\lceil \log_2(x) \rceil$-bit binary representation, and then when x is now a binary string, applies the Dirac operator as above.

- For a natural number $N \in \mathbb{N}$, the N-th complex root of unity is defined to be $\omega_N := e^{\frac{2i\pi}{N}}$.

- When we use the notation $\|\cdot\|$ for norm of a vector, we implicitly refer to the Euclidean, or ℓ_2 norm $\|\cdot\|_2$, unless explicitly noted otherwise.

- For two classical random variables D_0, D_1 over binary strings of finite length k, the total variation distance (sometimes referred to as statistical distance) between D_0 and D_1 is defined as follows:

$$\|D_0 - D_1\|_{TV} := \frac{1}{2} \|D_0 - D_1\|_1 := \frac{1}{2} \sum_{i \in [k], x \in \{0,1\}^i} |D_0(x) - D_1(x)| \ ,$$

where for $b \in \{0, 1\}$, $i \in [k]$ and $x \in \{0, 1\}^i$, the notation $D_b(x)$ is the probability to sample x in the distribution D_b.

- For two classical random variables D_0, D_1 over binary strings of finite length k, the fidelity between D_0 and D_1 is defined as follows:

$$F(D_0, D_1) := \sum_{i \in [k], x \in \{0,1\}^i} \sqrt{D_0(x) \cdot D_1(x)} \ ,$$

where for $b \in \{0, 1\}$, $i \in [k]$ and $x \in \{0, 1\}^i$, the notation $D_b(x)$ is the probability to sample x in the distribution D_b.

- For every pair of classical random variables D_0, D_1 over binary strings of finite length k, the following relation between the fidelity and total variation distance between distributions is known:

$$1 - F(D_0, D_1) \leq \|D_0 - D_1\|_{TV} \leq \sqrt{1 - F(D_0, D_1)^2} \ .$$

- Unless specifically noted otherwise, for any variable u (not necessarily denoting a classical string), the notation $|u\rangle$ denotes a unit vector in complex vector space, of some finite dimension. That is, the notation usually refers to a quantum state and is thus a normalized vector, unless explicitly noted otherwise. Also, for any $n \in \mathbb{N}$, an n-qubit state is always a unit vector in $\mathbb{C}^{2^n}$.

- A DPT algorithm is a classical deterministic polynomial-time Turing machine. It is a known fact in computational complexity theory that a DPT is equivalent to a uniform family of polynomial-sized classical deterministic circuits. We will use these two notions interchangeably throughout this work as our model for efficient classical computation.

- A PPT algorithm is a classical probabilistic polynomial-time Turing machine. It is a known fact in computational complexity theory that a PPT is equivalent to a uniform family of polynomial-sized classical probabilistic circuits. We will use these two notions interchangeably throughout this work as our model for efficient classical computation.

- A QPT algorithm is a quantum polynomial-time Turing machine. It is a known fact in computational complexity theory that a QPT is equivalent to a uniform family of polynomial-sized quantum circuits. We will use these two notions interchangeably throughout this work as our model for efficient quantum computation.

- Unitary transformations are the transformations which quantum mechanics allows, when excluding measurements (and adding ancilla qubits). Throughout this work we will consider unitaries as matrices, transformations or circuits, interchangeably, depending on the context, using the following conventions.

- For $n \in \mathbb{N}$, an n-qubit unitary is a unitary matrix $U \in \mathbb{C}^{2^n \times 2^n}$.

- For $n \in \mathbb{N}$, an n-qubit general circuit (or n-qubit quantum algorithm) is a quantum circuit C that can perform the constant-qubit quantum gates $\{X, Z, P, H, T, CNOT, SWAP, Tof\}$, execute measurement gates and add ancillary qubits registers containing $|0^k\rangle$.

- For $n \in \mathbb{N}$, an n-qubit unitary circuit is a quantum circuit on n qubits that only uses quantum gates $\{X, Z, P, H, T, CNOT, SWAP, Tof\}$, adds no ancilla and uses no measurement gates.

- For $n, k \in \mathbb{N}$, an n-qubit, k-ancilla unitary circuit C is a quantum circuit on n qubits that first concatenates a k-qubit ancilla $|0^k\rangle$ to the input, then executes only quantum gates $\{X, Z, P, H, T, CNOT, SWAP, Tof\}$. C satisfies that the ancillary qubits start and (crucially) return to 0^k, that is, there exists an n-qubit unitary U_C such that,

$$\text{For every } n\text{-qubit state } |\psi\rangle \ : C\left(|\psi\rangle|0^k\rangle\right) = (U|\psi\rangle)|0^k\rangle \ .$$

In that case, we say that the quantum circuit C implements the unitary U_C.

- For $n \in \mathbb{N}$ and an n-qubit unitary U, a unitary circuit for U, denoted C_U (or U, when it is clear from the context whether we are referring to the matrix or circuit version of U) is an n-qubit, k-ancilla (for some $k \in (\mathbb{N} \cup \{0\})$) quantum circuit that implements the unitary U.

- Representing permutations: For $m \in \mathbb{N}$, our way for presenting a permutation $\sigma \in S_m$ in the form of a binary string in this paper, is by using $m \cdot \lceil \log_2(m) \rceil$ bits, interpreted as a list of m numbers in $[m]$ – an ordering of the set $[m]$ according to the permutation σ. Specifically, the number in the i-th packet of $\lceil \log_2(m) \rceil$ bits, is $\sigma^{-1}(i)$.

- In this work we assume that quantum systems have finite dimensions. Formally, for every quantum register R of $n \in \mathbb{N}$ qubits we assume there exists a quantum register R' of finite size containing $n' \in \mathbb{N}$ qubits such that the quantum state of the joint system (R, R') is in a pure state, that is, the joint system is not entangled with any other outside quantum system.

- For natural numbers $N_0, N_1 \in \mathbb{N}$, the set $\mathbb{Z}_{(-N_0, N_1)}$ is defined as the set of integers between (and including) $-N_0$ and N_1, that is,

$$\mathbb{Z}_{(-N_0, N_1)} := \{-N_0, \cdots, -1, 0, 1, \cdots, N_1\} \ .$$

3.1 Standard Notions from Computational Complexity Theory

In this work we study generalization of decision problems, which are called promise problems. Promise problems are defined as follows.

Definition 3.1 (Promise Problem). *Let $\{0,1\}^* := \bigcup_{i \in (\mathbb{N} \cup \{0\})} \{0,1\}^i$ the set of all binary strings, of all lengths. A promise problem $\prod := (\prod_{YES}, \prod_{NO})$ is given by two (possibly infinite) sets $\prod_{YES} \subseteq \{0,1\}^*$, $\prod_{NO} \subseteq \{0,1\}^*$ such that $\prod_{YES} \cap \prod_{NO} = \emptyset$.*

A decision problem, which is a private case of promise problems, is defined as follows. It basically means a promise problem where the input can be any string, and is not promised to be in any particular set.

Definition 3.2 (Decision Problem). *Let $\prod := (\prod_{YES}, \prod_{NO})$ a promise problem. We say that $\prod$ is a decision problem if $\prod_{YES} \cup \prod_{NO} = \{0,1\}^*$.*

We next define languages, which are derived from decision problems.

Definition 3.3 (Language). *Let $\prod := (\prod_{YES}, \prod_{NO})$ a decision problem. We say that L is the language induced by the decision problem $\prod$ if $\prod_{YES} = L$.*

The complexity class **BQP** represents all computational problems that have a feasible solution on a quantum computer, and more broadly (and in short), **BQP** represents humanly-feasible quantum computation. The writing **BQP** stands for Bounded-Error Quantum Polynomial-Time, and defined as follows (which is one out of many equivalent definitions of this class).

Definition 3.4 (The Complexity Class **BQP**). *The complexity class **BQP** is the set of promise problems $\prod := (\prod_{YES}, \prod_{NO})$ that are solvable by a quantum polynomial-time algorithm. Formally, such that there exists a QPT M such that for every $x \in \prod_{YES}$, $M(x) = 1$ with probability at least $p(x)$, and if $x \in \prod_{NO}$, $M(x) = 0$ with probability at least $p(x)$, such that*

$$\forall x \in \left(\prod_{YES} \cup \prod_{NO} \right) : p(x) \geq 1 - 2^{-\mathsf{poly}(|x|)} \ ,$$

for some polynomial $\mathsf{poly}(\cdot)$.

We next discuss computational problems (that is, promise problems) on arbitrary mathematical objects. As in the standard conventions in computational complexity theory, arbitrary mathematical objects can be represented by binary strings, and thus promise problems capture any problem.

The Statistical Difference Problem. We next define the Statistical Difference (SD) problem [SV03], a well-known promise problem in computer science.

Definition 3.5 (The Statistical Difference Problem). *Let $a, b : \mathbb{N} \rightarrow [0, 1]$ functions such that $\forall n \in \mathbb{N} : a(n) \leq b(n)$. The $(a(n), b(n))$-gap Statistical Difference problem, denoted $\mathsf{SD}_{a(n),b(n)}$ is a promise problem, where the input is a pair of classical circuits (C_0, C_1) with the same output size, $C_0 : \{0,1\}^{k_0} \rightarrow \{0,1\}^k$, $C_1 : \{0,1\}^{k_1} \rightarrow \{0,1\}^k$. For a pair of circuits we define their output distributions:*

$$\forall b \in \{0,1\} : D_b := \{C_b(x) \,|\, x \leftarrow \{0,1\}^{k_b}\} \ .$$

The problem is defined as follows.

- $\prod_{YES}$ *is the set of pairs of circuits (C_0, C_1) such that $\|D_0 - D_1\|_{TV} \leq a(n)$,*

- $\prod_{NO}$ *is the set of pairs of circuits (C_0, C_1) such that $\|D_0 - D_1\|_{TV} > b(n)$,*

such that n is the description size of the pair of circuits,

$$n := |(C_0, C_1)| \ .$$

In this work we mainly care about the statistical difference problem where the gap is not too small, and formally, lower-bounded inverse-polynomially. To this end we define the *family* of problems $\mathsf{SD}_{\mathsf{poly}}$ as follows.

Definition 3.6 (The Polynomial-Gap Statistical Difference Problem). *The polynomially-bounded statistical difference problem, denoted $\mathsf{SD}_{\mathsf{poly}}$ is in fact a set of promise problems (and not a promise problem by itself). It is defined as follows.*

$$\mathsf{SD}_{\mathsf{poly}} := \bigcup_{\substack{a(n),b(n):\mathbb{N}\rightarrow[0,1], \\ \exists \text{ a polynomial } \mathsf{poly}:\mathbb{N}\rightarrow\mathbb{N} \text{ such that } \forall n\in\mathbb{N}:b(n)^2-2a(n)+a(n)^2\geq\frac{1}{\mathsf{poly}(n)}}} \{\mathsf{SD}_{a(n),b(n)}\} \ .$$

The Graph Isomorphism Problem. As part of this work we show how a strengthening of the model of quantum computation solves the graph isomorphism problem. To this end, we define (simple and undirected) graphs and then the graph isomorphism problem. First, a (simple and undirected) graph with n vertices is simply a set of n points, implicitly numbered $[n]$, where some pairs of points have edges between them. The edges are are lines, or connections, between a pair of points $i, j \in [n]$, $i \neq j$. In this work when we refer to graphs, they are implicitly simple and undirected.

Definition 3.7 (Simple and Undirected Graph). *For $n \in \mathbb{N}$, an n-vertex (simple and undirected) graph G is given by the pair $([n], E)$, such that E contains size-2 subsets of $[n]$.*

Observe that the maximal amount of edges, that is, the maximal size of E for an n-vertex graph G, is $\binom{n}{2} := \frac{n(n-1)}{2}$. This means that a graph can be represented by a binary string of length $O(n^2)$. Next, we define isomorphic graphs. The intuition behind graph isomorphism is that isomorphic graphs G_0, G_1 are essentially the exact same graph (they contain the same information on connections), and visually, by moving the vertices of (any embedding of) one of the graphs G_b, we get a picture of an embedding of the other $G_{\neg b}$.

Definition 3.8 (Isomorphic Graphs). *Let $n \in \mathbb{N}$ and let $G_0 = ([n], E_0)$, $G_1 = ([n], E_1)$ a pair of n-vertex graphs. We say that G_0 and G_1 are isomorphic and denote $G_0 \simeq G_1$ if there exists a bijection (or equivalently, permutation) f on the set $[n]$ such that,*

$$\forall i, j \in [n] : \{i, j\} \in E_0 \iff \{f(i), f(j)\} \in E_1 \ .$$

The Graph Isomorphism (GI) problem is the computational problem of deciding whether a pair of input graphs are isomorphic or not. The GI problem has a long history of attempted solutions using polynomial-time algorithms, both classically and quantumly.

Definition 3.9 (The Graph Isomorphism Problem). *The Graph Isomorphism problem, denoted GI is a decision problem, where the input is a pair of graphs (G_0, G_1) with the same number of vertices n. The problem is defined as follows.*

- $\prod_{YES}$ *is the set of pairs of graphs (G_0, G_1) such that $G_0 \simeq G_1$.*

- $\prod_{NO}$ *is $\{0, 1\}^* \setminus \prod_{YES}$.*

As a final comment, the GI is a decision problem (and not only the more general, promise problem) since if it had a promise, this promise would turn out to be computationally trivial. That is, the set $\prod_{\text{NO}}$ is the sets of strings that either (1) just don't represent a pair of graphs, or (2) represent a pair of graphs (G_0, G_1), but this pair is not isomorphic. Checking whether an input string has the correct format and indeed represents a pair of graphs can be done in classical polynomial-time, and thus the graph isomorphism problem is usually thought of as a decision problem and not just a promise problem.

We use the following known facts about the distributions of random permutations of isomorphic graphs, and also on known algorithmic techniques for sampling a uniformly random permutation.

Fact 3.1 (Permutations of isomorphic and non-isomorphic graphs). *Let (G_0, G_1) a pair of graphs on $n \in \mathbb{N}$ vertices each.*

In case the pair are isomorphic, then for a permutation $\sigma \in S_n$ on the set $[n]$, denote by $\sigma(G)$ the graph generated by rearranging the vertices of G with accordance to the permutation σ. then, the following distributions are identical:

$$\{\sigma(G_0) \,|\, \sigma \leftarrow S_n\} \equiv \{\sigma(G_1) \,|\, \sigma \leftarrow S_n\} \ .$$

In case the pair of graphs is not isomorphic, then the above distributions have disjoint supports.

Fact 3.2 (The Fisher-Yates permutation-sampling algorithm [FY53]). *Let $n \in \mathbb{N}$, the following process produces a uniformly random permutation $\sigma \in S_n$. Start with any permutation of $(1, 2, \cdots, n-1, n)$, and for a decreasing $i = n, n-1, \cdots, 3, 2$, for every i, make a swap between the element currently in slot i and the element currently in slot r_i, where r_i is a uniformly random element in $[i]$.*

The Gap Closest Vector Problem. We turn to the definition of the gap-closest vector problem (GapCVP). We define lattices and then the GapCVP problem.

For a basis $\mathbf{B}$ of $\mathbb{R}^n$, a lattice with basis $\mathbf{B}$ is the infinite set of all *integer* linear combinations of the basis vectors in $\mathbf{B}$. In this work, in order to not deal with real-values vectors and finite precision, we consider only lattice bases with integer coordinates, i.e., bases of $\mathbb{R}^n$ from $\mathbb{Z}^n$. The computational hardness of these problems are polynomially equivalent.

Definition 3.10 (An Integer Lattice). *Let $n \in \mathbb{N}$ and let $\mathbf{B} := \{\mathbf{b}_1, \mathbf{b}_2, \cdots, \mathbf{b}_n\}$ a basis for $\mathbb{R}^n$ with integer vectors, that is, $\mathbf{B} \subseteq \mathbb{Z}^n$. The lattice of $\mathbf{B}$, denoted $\mathcal{L}_{\mathbf{B}}$, is defined as follows.*

$$\mathcal{L}_{\mathbf{B}} := \left\{ \mathbf{a} \in \mathbb{Z}^n \,\middle|\, \exists \mathbf{s} \in \mathbb{Z}^n : \mathbf{a} = \mathbf{B} \cdot \mathbf{s} \right\} \ .$$

For a lattice $\mathcal{L}$ and a vector $\mathbf{t} \in \mathbb{Z}^n$, the distance $\Delta(\mathcal{L}, \mathbf{t})$ between $\mathcal{L}$ and $\mathbf{t}$ is the minimum (Euclidean, ℓ_2) distance between $\mathbf{v}$ and $\mathbf{t}$, where the minimum is taken over $\mathbf{v} \in \mathcal{L}$. Next, we define the Gap Closest Vector Problem.

Definition 3.11 (The Gap Closest Vector Problem). *Let $g : \mathbb{N} \to \mathbb{R}_{\geq 1}$ a function. The $g(n)$-Gap Closest Vector Problem, denoted $\mathsf{GapCVP}_{g(n)}$ is a promise problem, where the input is a triplet $(\mathbf{B}, \mathbf{t}, d)$ such that $\mathbf{B} \subseteq \mathbb{Z}^n$ is a basis for $\mathbb{R}^n$, $\mathbf{t} \in \mathbb{Z}^n$ is an integer vector and $d \in \mathbb{N}$. For an input triplet, the promise problem is defined as follows.*

- *$\prod_{YES}$ is the set of triplets $(\mathbf{B}, \mathbf{t}, d)$ such that $\Delta(\mathcal{L}_\mathbf{B}, \mathbf{t}) \leq d$.*

- *$\prod_{NO}$ is the set of triplets $(\mathbf{B}, \mathbf{t}, d)$ such that $\Delta(\mathcal{L}_\mathbf{B}, \mathbf{t}) > d \cdot g(n)$.*

3.2 Statistical and Algorithmic Tools

We use numerous algorithmic and statistical results in this work.

Quantum Swap Test. Throughout this work we use the Swap Test [BBD+97, BCWDW01] algorithm, which, among other things, can be used to distinguish between quantum states as a function of their inner product as vectors.

Theorem 3.1 (Swap Test Algorithm). *There exists a quantum algorithm $Q_{ST}(\cdot)$ that gets as input a $2 \cdot n$-qubit register R and outputs a bit $b \in \{0, 1\}$, and executes in complexity $O(n)$. If R is in the separable state of n qubits each $|\psi\rangle \otimes |\phi\rangle$, then the algorithm outputs 1 with probability $\frac{1}{2} + \frac{|\langle\psi||\phi\rangle|^2}{2}$, and with the remaining probability outputs 0.*

Deviation Bounds. We use the following known formulation of Chernoff's bound, which shows an exponentially fast (in the number of samples) decaying probability for the deviation of the average of many binary i.i.d. samples. One of the main usages of the below statement in repetition theorems in computer science, is that the repetition parameter $n \in \mathbb{N}$ in the below statement can be chosen as a function of both $\varepsilon \in [0, 1]$ and $p \in [0, 1]$.

Theorem 3.2 (Chernoff's Bound for an Average of i.i.d. Samples). *Let $n \in \mathbb{N}$, let X a random variable over $\{0, 1\}$ (that is, $X = 1$ with probability $p \in [0, 1]$ and $X = 0$ with probability $1 - p$) and let $\{X_i\}_{i \in [n]}$ n i.i.d. samples from X. Let $A_{X,n} := \frac{\sum_{i \in [n]} X_i}{n}$ the average of the sum of samples, and let $E(A_{X,n})$ the expectation of the average. Then for every $\varepsilon \in (0, 1)$,*

$$\Pr_{X_1, \cdots, X_n} \left[\left| A_{X,n} - E(A_{X,n}) \right| \geq \varepsilon \cdot p \right] \leq 2 \cdot e^{-n \cdot \frac{p \cdot \varepsilon^2}{3}} .$$

Gaussian distribution variants. As part of the reduction $\mathsf{GapCVP} \leq_p \mathsf{SISD}_{\mathsf{poly},1}$, which is resented in Section 7, we will approximate the discrete truncated Gaussian distribution, which is defined as follows.

Definition 3.12 (Discrete truncated Gaussian distribution). *For $B \in \mathbb{N}$, we define the discrete truncated Gaussian distribution D_B as the distribution with sample space $\{-B, \cdots, -1, 0, 1, \cdots, B\}$ defined as,*

$$\forall x \in \{-B, \cdots, -1, 0, 1, \cdots, B\} :$$

$$D_B(x) := \frac{e^{-\frac{\pi \cdot |x|^2}{B^2}}}{\sum_{y \in \{-B, \cdots, -1, 0, 1, \cdots, B\}} e^{-\frac{\pi \cdot |y|^2}{B^2}}} .$$

As part of the approximation of the discrete Gaussian distribution we will use a closely related variant, which is the *rounded* truncated Gaussian distribution. To this end we first define the continuous Gaussian distribution, then rounded Gaussian distribution, and then the truncated version of the rounded distribution.

Definition 3.13 (Continuous Gaussian distribution). *Let $\mu \in \mathbb{R}$ and $\sigma \in \mathbb{R}_{\geq 0}$. The continuous Gaussian (or normal) distribution with mean μ and standard deviation σ, denoted $\mathcal{N}(\mu, \sigma)$ is the distribution with sample space $\mathbb{R}$ such that,*

$$\forall x \in \mathbb{R} : \mathcal{N}(\mu, \sigma)(x) := \frac{1}{\sigma \sqrt{2\pi}} \cdot e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2} \ .$$

Definition 3.14 (Rounded Gaussian distribution). *Let $\mu \in \mathbb{R}$ and $\sigma \in \mathbb{R}_{\geq 0}$. The rounded Gaussian distribution with mean μ and standard deviation σ, denoted $\mathcal{N}^{\mathbb{Z}}(\mu, \sigma)$ is the distribution with sample space $\mathbb{Z}$ such that,*

$$\forall x \in \mathbb{Z} : \mathcal{N}^{\mathbb{Z}}(\mu, \sigma)(x) := \int_{x-\frac{1}{2}}^{x+\frac{1}{2}} \mathcal{N}(\mu, \sigma)(x) \ ,$$

where $\mathcal{N}(\mu, \sigma)$ is the probability density function of the continuous Gaussian distribution (Definition 3.13) with mean μ and standard deviation σ.

Definition 3.15 (Rounded truncated Gaussian distribution). *Let $\mu \in \mathbb{R}$, $\sigma \in \mathbb{R}_{\geq 0}$ and $B \in \mathbb{N}$. The rounded B-truncated Gaussian distribution with mean μ and standard deviation σ, denoted $\mathcal{N}^{B}(\mu, \sigma)$ is the distribution with sample space $\{-B, \cdots, -1, 0, 1, \cdots, B\}$ such that,*

$$\forall x \in \{-B, \cdots, -1, 0, 1, \cdots, B\} :$$

$$\mathcal{N}^{B}(\mu, \sigma)(x) := \frac{\mathcal{N}^{\mathbb{Z}}(\mu, \sigma)(x)}{\sum_{y \in \{-B, \cdots, 1, 0, 1, \cdots, B\}} \mathcal{N}^{\mathbb{Z}}(\mu, \sigma)(y)} \ ,$$

where $\mathcal{N}^{\mathbb{Z}}(\mu, \sigma)$ is the probability density function of the rounded Gaussian distribution (Definition 3.14) with mean μ and standard deviation σ.

We use the following fact on the total variation distance between the two versions (discrete and rounded) of the truncated Gaussian distributions.

Fact 3.3 (Distance between rounded and discrete Gaussians). *Let $B \in \mathbb{N}$, then,*

$$\left\| \left(\mathcal{N}^{B}(0, B)\right) - (D_B) \right\|_{TV} \leq O\left(\frac{1}{B}\right) \ .$$

Strengthening of the central limit theorem. The central limit theorem (CLT), one of the fundamental theorems in statistics, tells us that when we take $N \in \mathbb{N}$ independent and continuous random variables $X_N, \cdots, X_N$, the sum of the samples approaches the Gaussian distribution, as N goes to infinity. The CLT does not tell us the rate at which the samples approach the Gaussian distribution, as this depends, among other things, on the probability density functions of the summed variables $\{X_i\}_{i \in [N]}$. The Berry-Esseen Theorem is a quantified version of the central limit theorem – it bounds the Wasserstein-1 distance between the sum $S := \sum_{i \in [N]} X_i$ and the Gaussian distribution, as a function of N and the density functions of the variables $\{X_i\}_{i \in [N]}$. The original formulation of the Berry-Essen Theorem is insufficient for our needs, because (1) In this paper we are working with discrete random variables and not continuous, and (2) We want a bound on the total variation distance which implies physical statistical indistinguishability, and for this, the Wasserstein-1 distance is not enough. We will use the following strengthening Theorem from [CGS10]:

Theorem 3.3 (Discrete Berry-Esseen bound for total variation – Theorem 7.4 from [CGS10]). *Let $N \in \mathbb{N}$ and let $\{X_i\}_{i \in [N]}$ independent random variables with finite supports contained in $\mathbb{Z}$, and denote $S := \sum_{i \in [N]} X_i$. Denote,*

- *For every $i \in [N]$, the expectation of X_i, $\mu_i := \mathbb{E}(X_i)$.*

- *For every $i \in [N]$, the variance of X_i, $\sigma_i^2 := Var(X_i) := \mathbb{E}(X_i^2) - \mathbb{E}(X_i)^2$.*

- *The variance of the sum S, $\sigma^2 := Var(S)$, which equals $\sum_{i \in [N]} \sigma_i^2$, due to the random variables being independent.*

- *For every $i \in [N]$, the third moment of X_i, $\gamma_i := \frac{\mathbb{E}_{X_i}(|X_i - \mu_i|^3)}{\sigma^3}$.*

- *The expectation of S, $\mu := \sum_{i \in [N]} \mu_i$.*

- *The sum of third moments $\gamma := \sum_{i \in [N]} \gamma_i$.*

- *The sum of the maximal values that the variables can take, $B := \sum_{i \in [N]} (\max_{x \in X_i} |x|)$.*

- *For every $i \in [N]$, the random variable $S^{(i)} := S - X_i$, that is, the original full sum S, excluding the i-th variable.*

Then we have the following upper bound on the total variation distance,

$$\left\| S - \mathcal{N}^B(\mu, \sigma) \right\|_{TV}$$

$$\leq \frac{3}{2} \cdot \sigma \cdot \sum_{i \in [N]} \left(\left(\gamma_i + \frac{2 \cdot \sigma_i^2}{3 \cdot \sigma^3} \right) \cdot \left\| \left(S^{(i)} \right) - \left(S^{(i)} + 1 \right) \right\|_{TV} \right)$$

$$+ \left(5 + 3\sqrt{\frac{\pi}{8}} \right) \cdot \gamma + \frac{1}{\sigma \cdot 2\sqrt{2\pi}} \ .$$

4 New Notions and Definitions

In this section we define the new notions that this work presents. We start with the computational model and then a comparison to standard quantum computation. We then proceed to defining sequentially invertible circuits and the sequentially invertible statistical difference problem SISD.

4.1 Quantum Computation with Computable Order Interference

At the heart of our model is the order interference (OI) oracle, which gets as input an amplification parameter $\lambda \in \mathbb{N}$, a set of $m \in \mathbb{N}$ unitaries $\{U_i\}_{i \in [m]}$ (all acting on the same number of qubits, $n \in \mathbb{N}$) and an n-qubit state $|\psi\rangle$. The oracle then makes an attempt to execute an OI transformation, and if the OI transformation succeeds, we get the OI state. The relevant definitions follow.

Definition 4.1 (The Uniform Order Interference State). *Let $\{U_i\}_{i \in [m]}$ a set of $m \in \mathbb{N}$ unitary circuits which all operate on the same number of qubits $n \in \mathbb{N}$ and let $|\psi\rangle$ an n-qubit quantum state.*

The uniform order interference state of the pair $\left(\{U_i\}_{i \in [m]}, |\psi\rangle \right)$ is

$$\boldsymbol{OI}\left(\{U_i\}_{i \in [m]}, |\psi\rangle \right) := \sum_{\sigma \in S_m} \left(\prod_{i \in [m]} U_{\sigma^{-1}(i)} \right) \cdot |\psi\rangle \ .$$

After defining the uniform OI state, we define the uniform OI oracle as follows.

Definition 4.2 (Computable Uniform Order Interference Oracle)**.** *The computable uniform order inter-ference oracle, denoted $\mathcal{O}_{OI}$, is an oracle with input triplet $\left(\{U_i\}_{i\in[m]}, R, \lambda\right)$, such that,*

- *$\{U_i\}_{i\in[m]}$ is a set of $m \in \mathbb{N}$ unitary circuits which all operate on the same number of qubits $n \in \mathbb{N}$,*

- *R is an n-qubit quantum register,*

- *and $\lambda \in \mathbb{N}$ is a natural number.*

Given valid input $\left(\{U_i\}_{i\in[m]}, R, \lambda\right)$, let R' a n'-qubit register such that the joint system (R, R') is in a pure state, written as,

$$\sum_{x\in\{0,1\}^{n'}} \alpha_x \cdot |\psi_x\rangle_R |x\rangle_{R'} \ .$$

The oracle call takes time complexity $\sum_{i\in[m]} |U_i| + \lambda$. At the end of the oracle execution it returns a success/failure bit $b \in \{0, 1\}$, where the success probability is

$$\min_{x\in\{0,1\}^{n'}} \left(\frac{\frac{\left\|\boldsymbol{OI}\left(\{U_i\}_{i\in[m]}, |\psi_x\rangle\right)\right\|}{m!}}{\frac{\left\|\boldsymbol{OI}\left(\{U_i\}_{i\in[m]}, |\psi_x\rangle\right)\right\|}{m!} + \frac{1}{\lambda}} \right) \ .$$

If the call succeeded, the state in (R, R') is the normalization of,

$$\sum_{x\in\{0,1\}^{k'}} \alpha_x \cdot \boldsymbol{OI}\left(\{U_i\}_{i\in[m]}, |\psi_x\rangle\right)_R |x\rangle_{R'} \ ,$$

and if it failed, the state in (R, R') may be arbitrary.

To make using an OI oracle easier, when the n-qubit input register R to the oracle is in a pure state, the outcome of an oracle access is simplified.

Claim 4.1 (Corollary - Uniform OI on pure states)**.** *Let $\left(\{U_i\}_{i\in[m]}, R, \lambda\right)$ such that $\{U_i\}_{i\in[m]}$ is a set of $m \in \mathbb{N}$ unitary circuits which all operate on the same number of qubits $n \in \mathbb{N}$, $\lambda \in \mathbb{N}$ is a natural number and R is an n-qubit register in some pure state $|\psi\rangle$.*
Then, the oracle call $\mathcal{O}_{OI}\left(\{U_i\}_{i\in[m]}, R, \lambda\right)$ succeeds with probability,

$$\frac{\frac{\left\|\boldsymbol{OI}\left(\{U_i\}_{i\in[m]}, |\psi\rangle\right)\right\|}{m!}}{\frac{\left\|\boldsymbol{OI}\left(\{U_i\}_{i\in[m]}, |\psi\rangle\right)\right\|}{m!} + \frac{1}{\lambda}} \ ,$$

and with (at least) the same probability, the state in R after the oracle call is the normalization of the uniform OI state, $\boldsymbol{OI}\left(\{U_i\}_{i\in[m]}, |\psi\rangle\right)$.

General form of OI states and oracles. While this work studies only the the computational power of a quantum computer having the ability to generate uniform superpositions of unitary execution orders, it is an interesting mathematical question for future work to understand what's possible to compute when the superposition of orders is not necessarily uniform. To this end, we next define the generalized OI state and generalized OI oracle. Both of the latter have a fourth parameter U_O, a unitary circuit which determines the superposition of orders. We are overloading notations and still use **OI**, such that when we are dropping the parameter for U_O, we are automatically referring to the uniform definitions of OI state and OI oracle.

Definition 4.3 (The Order Interference State). *Let $\{U_i\}_{i\in[m]}$ a set of $m \in \mathbb{N}$ unitary circuits which all operate on the same number of qubits $n \in \mathbb{N}$, let $|\psi\rangle$ an n-qubit quantum state, and let U_O a k-qubit unitary circuit on registers (O, A) where O has $m \cdot \lceil \log(m) \rceil$ qubits and A has the rest.*

Write the generated state by U_O in the following way (recall that we use $m \cdot \lceil \log(m) \rceil$ bits to represent a permutation in S_m):

$$U_O \cdot |0^k\rangle = \sum_{\sigma \in S_m} \alpha_\sigma \cdot |\sigma\rangle_O |\psi_\sigma\rangle_A \ .$$

The order interference state of the triplet $\left(\{U_i\}_{i\in[m]}, |\psi\rangle, U_O\right)$ is defined as follows.

$$\boldsymbol{OI}\left(\{U_i\}_{i\in[m]}, |\psi\rangle, U_O\right) := \sum_{\sigma \in S_m} \alpha_\sigma \cdot \left(\prod_{i\in[m]} U_{\sigma^{-1}(i)}\right) \cdot |\psi\rangle \ .$$

- When the notation $\boldsymbol{OI}(\cdot)$ is used with 2 parameters instead of 3, we automatically refer to the uniform OI state, as in definition 4.1.

We next define the generalized OI oracle. One thing to notice is that unlike in the case of the uniform OI state, the general OI state is such that amplitudes are taken into consideration. The consideration makes a shorter vector. This is taken into consideration when calculating the norm of the general OI state, which is relevant for the success probability of an oracle call. Formally, this pops up in the below definition of a general OI oracle, where the success probability of the call scales down the OI state by a factor of $\sqrt{m!}$ rather than by a factor of $m!$, as in the uniform case.

Definition 4.4 (Computable Order Interference Oracle). *A computable order interference oracle, denoted $\mathcal{O}_{\boldsymbol{OI}}$, is an oracle with input quadruple $\left(\{U_i\}_{i\in[m]}, R, U_O, \lambda\right)$, such that,*

- $\{U_i\}_{i\in[m]}$ *is a set of $m \in \mathbb{N}$ unitary circuits which all operate on the same number of qubits $n \in \mathbb{N}$,*

- R *is an n-qubit quantum register,*

- U_O *is a unitary on registers (O, A) where O has $m \cdot \lceil \log(m) \rceil$ qubits and A has some number of qubits $k \in \mathbb{N}$,*

- *and $\lambda \in \mathbb{N}$ is a natural number.*

Given valid input $\left(\{U_i\}_{i\in[m]}, R, U_O, \lambda\right)$, let R' a n'-qubit register such that the joint system (R, R') is in a pure state, written as,

$$\sum_{x\in\{0,1\}^{n'}} \alpha_x \cdot |\psi_x\rangle_R |x\rangle_{R'} \ .$$

The oracle call takes time complexity $|U_O| + \sum_{i\in[m]} |U_i| + \lambda$. At the end of the oracle execution it returns a success/failure bit $b \in \{0, 1\}$, where the success probability is

$$\min_{x\in\{0,1\}^{n'}} \left(\frac{\frac{\left\| \boldsymbol{OI}(\{U_i\}_{i\in[m]}, |\psi_x\rangle, U_O) \right\|}{\sqrt{m!}}}{\frac{\left\| \boldsymbol{OI}(\{U_i\}_{i\in[m]}, |\psi_x\rangle, U_O) \right\|}{\sqrt{m!}} + \frac{1}{\lambda}} \right) \ .$$

If the call succeeded, the state in (R, R') is the normalization of,

$$\sum_{x\in\{0,1\}^{n'}} \alpha_x \cdot \boldsymbol{OI}\left(\{U_i\}_{i\in[m]}, |\psi_x\rangle, U_O\right)_R |x\rangle_{R'} \ ,$$

and if it failed, the state in (R, R') may be arbitrary.

When the notation of an oracle call $\mathcal{O}_{\boldsymbol{OI}}(\cdot)$ is used with 3 parameters instead of 4, we automatically refer to the computable uniform OI oracle, as in definition 4.2.

Remark 4.1 (General computable OI implies uniform computable OI). The explanation for why the uniform versions of the OI notions are indeed private cases of the general cases, and formally, why the general computable OI oracle can simulate the uniform OI oracle, is as follows. By using the unitary U_R from Claim 4.2, followed by the unitary U_{FY} from Claim 4.3, we have a unitary U_{uni} on $2 \cdot m \cdot \lceil \log(m) \rceil$ qubits (possibly using extra ancillary qubits) that generates,

$$U_{uni} \cdot |0^{2 \cdot m \cdot \lceil \log(m) \rceil}\rangle = \frac{1}{\sqrt{m!}} \sum_{\sigma \in S_m} |r_\sigma\rangle |\sigma\rangle \ .$$

More elaboration on this is given as part of the proof of Lemma 4.2. Thus, in order to apply uniform OI transformations using the general OI oracle, we execute a call to the general OI oracle with $U_O := U_{uni}$.

Throughout we use the following lemma, which compiles the definition of a uniform OI oracle and the needed amplification parameter to make the success probability arbitrarily close to 1.

Lemma 4.1 (Amplification of Success Probability of OI Transformations). *Let $n, m \in \mathbb{N}$, let $\{U_i\}_{i \in [m]}$ a set of n-qubit unitaries, and let $|\psi\rangle$ an n-qubit quantum state. Let $k \in \mathbb{N}$ and let*

$$\lambda := \left\lceil k \cdot \frac{m!}{\left\| \boldsymbol{OI}\left(\{U_i\}_{i \in [m]}, |\psi\rangle\right) \right\|} \right\rceil \ .$$

Then, the success probability of the oracle call $\mathcal{O}_{\boldsymbol{OI}}\left(\lambda, \{U_i\}_{i \in [m]}, |\psi\rangle\right)$ is $\geq 1 - \frac{1}{k}$.

Proof. To prove the claim, all that is needed is to calculate the success probability, with accordance to the definition of the OI oracle.

$$\frac{\frac{\left\| \boldsymbol{OI}(\{U_i\}_{i \in [m]}, |\psi\rangle) \right\|}{m!}}{\frac{\left\| \boldsymbol{OI}(\{U_i\}_{i \in [m]}, |\psi\rangle) \right\|}{m!} + \frac{1}{\lambda}}$$

$$= \frac{\frac{\left\| \boldsymbol{OI}(\{U_i\}_{i \in [m]}, |\psi\rangle) \right\|}{m!} + \frac{1}{\lambda} - \frac{1}{\lambda}}{\frac{\left\| \boldsymbol{OI}(\{U_i\}_{i \in [m]}, |\psi\rangle) \right\|}{m!} + \frac{1}{\lambda}}$$

$$= 1 - \frac{\frac{1}{\lambda}}{\frac{\left\| \boldsymbol{OI}(\{U_i\}_{i \in [m]}, |\psi\rangle) \right\|}{m!} + \frac{1}{\lambda}}$$

$$= 1 - \frac{1}{\lambda \cdot \frac{\left\| \boldsymbol{OI}(\{U_i\}_{i \in [m]}, |\psi\rangle) \right\|}{m!} + 1}$$

$$\geq 1 - \frac{1}{\lambda \cdot \frac{\left\| \boldsymbol{OI}(\{U_i\}_{i \in [m]}, |\psi\rangle) \right\|}{m!}}$$

$$\underset{\text{(Definition of } \lambda)}{\geq} 1 - \frac{1}{k} \ .$$

$\square$

Quantum Computation with Order Interference - Complexity Class. We give the definition of the set of computational problems solvable in polynomial time, by a quantum computer with the ability to create superposition of unitary execution orders. The class $\mathbf{BQP^{OI}}$ capturing the computational power of a polynomial-time quantum computer with access to a (generalized) OI oracle.

Definition 4.5 (The Complexity Class $\mathbf{BQP^{OI}}$). *The complexity class $\boldsymbol{BQP^{OI}}$ is the set of promise problems $\prod := \left(\prod_{YES}, \prod_{NO}\right)$ such that there exists $M^{\boldsymbol{OI}}$ a QPT with access to an OI oracle $\mathcal{O}_{\boldsymbol{OI}}$ (as*

in Definition 4.4), such that for every $x \in \prod_{YES}$, $M^{OI}(x) = 1$ with probability at least $p(x)$, and if $x \in \prod_{NO}$, $M^{OI}(x) = 0$ with probability at least $p(x)$, such that

$$\forall x \in \left(\prod_{YES} \cup \prod_{NO} \right) : p(x) \geq 1 - 2^{-\mathsf{poly}(|x|)} \ ,$$

for some polynomial $\mathsf{poly}(\cdot)$.

4.2 Comparison to Standard Quantum Computation

In this section we give a brief comparison between quantum computation with and without computable OI. Specifically, we show techniques in (standard) quantum computation to simulate a computable uniform OI oracle. First, we show a quantum polynomial-time algorithm for the generation of an entangled uniform superposition of permutation states.

Claim 4.2 (Factorial transformation). *For every $m \in \mathbb{N}$ there exists a polynomial size (in m) unitary circuit U_R on $m \cdot \lceil \log(m) \rceil$ qubits (possibly using ancillary qubits) that maps*

$$\forall x_m \in [m], x_{m-1} \in [m-1], \cdots, x_2 \in [2] : U_R \cdot |x_m, x_{m-1}, \cdots, x_2\rangle =$$

$$\sum_{i_m \in [m], i_{m-1} \in [m-1], \cdots, i_2 \in [2]} \frac{1}{\sqrt{m!}} \left(\prod_{j \in [m]} \omega_j^{x_j \cdot i_j} \right) |i_m, i_{m-1}, \cdots, i_2\rangle$$

Proof. The proof follows readily by recalling the ability to execute the quantum Fourier transform QFT_N for any number N (where N is not necessarily a power of 2), using a $\lceil \log_2(N) \rceil$-qubit unitary quantum circuit $U_{\mathsf{QFT},N}$ (which possibly uses ancillary qubits) [Kit95, MZ04] that executes in polynomial time in $\log_2(N)$.

Finally, by executing in parallel, m Fourier transforms of different orders, that is, the i-th transform is QFT_i, we get our wanted transformation. Formally, define $U_R := \bigotimes_{j=m,m-1,\cdots,3,2} U_{\mathsf{QFT},j}$, and we get,

$$U_R \cdot |x_m, x_{m-1}, \cdots, x_2\rangle = \bigotimes_{j=m,m-1,\cdots,3,2} U_{\mathsf{QFT},j} \cdot |x_j\rangle$$

$$= \bigotimes_{j=m,m-1,\cdots,3,2} \left(\sum_{i_j \in [j]} \omega_j^{x_j \cdot i_j} \frac{1}{\sqrt{j}} |i_j\rangle \right)$$

$$= \sum_{i_m \in [m], i_{m-1} \in [m-1], \cdots, i_2 \in [2]} \left(\bigotimes_{j=m,m-1,\cdots,3,2} \frac{1}{\sqrt{j}} \omega_j^{x_j \cdot i_j} |i_j\rangle \right)$$

$$= \sum_{i_m \in [m], i_{m-1} \in [m-1], \cdots, i_2 \in [2]} \frac{1}{\sqrt{m!}} \left(\prod_{j \in [m]} \omega_j^{x_j \cdot i_j} \right) |i_m, i_{m-1}, \cdots, i_2\rangle \ ,$$

as needed. $\qquad \square$

We next state a claim, part of the proof for which, is part of the proof of correctness of the Fisher-Yates sampling algorithm. The FY algorithm samples a uniformly random permutation $\sigma \in S_m$ by having access to randomness of specific structure.

Claim 4.3 (Fisher-Yates Permutation Sampler). *Let $m \in \mathbb{N}$, let $R_m := [m] \times [m-1] \times \cdots \times [2]$ and let S_m the set of all permutations on $[m]$. An element from each of the sets R_m, S_m can be represented by $m \cdot \lceil \log(m) \rceil$ bits.*

- *There exists a bijection $B_{FY} : R_m \to S_m$. Note that this means that a uniformly random input from R_m produces a uniformly random permutation from S_m.*

- *There exists a polynomial size (in m) unitary circuit U_{FY} on $2 \cdot m \cdot \lceil \log(m) \rceil$ qubits (possibly using extra ancillary qubits), such that,*

$$\forall r \in R_m, y \in \{0,1\}^{m \cdot \lceil \log(m) \rceil} : U_{FY} \cdot |r, y\rangle = |r, y \oplus B_{FY}(r)\rangle \ .$$

Proof. The bijection B_{FY} is the mapping from the Fisher-Yates [FY53] algorithm. Basically, B_{FY} operates as follows: It starts from the identity permutation: $\sigma_I := (1, 2, 3, \cdots, m)$, which keeps every element in its place. Then, for every $i = 1, 2, \cdots m-1$, take r_i the i-th coordinate in r, and swap element 1 with element r_i.

Finally, the unitary U_{FY} follows as a standard fact in quantum computing: Since the function B_{FY} is classically efficiently computable, there is a unitary that computes it in particular. $\square$

Given the correspondence between elements $r \in R_m$ and permutations $\sigma \in S_m$, for $r \in R_m$ we denote $\sigma_r := B_{FY}(r)$, and for $\sigma \in S_m$ we denote $r_\sigma := B_{FY}^{-1}(\sigma)$. Next, we state without proof a trivial fact in quantum computation, namely, that if we are given (even only an oracle) access to controlled versions of unitaries $\{U_i\}_{i \in [m]}$ and any superposition of permutations over the set S_m, we can generate an *entangled* superposition of unitary execution orders, where the entanglement is between the order register (holding the order information, in the form of the permutation $\sigma \in S_m$) and the target register (holding the result of the chosen order of execution).

Fact 4.1. *For every $m, n \in \mathbb{N}$, there exists a polynomial size (in m, n) unitary circuit U_O on $m \cdot \lceil \log(m) \rceil + n$ qubits (possibly using extra ancillary qubits), that for every set of m unitaries $\{U_i\}_{i \in [m]}$ all act on the same number of qubits n, given oracle access to controlled versions of the unitaries in the set, executes the following unitary transformation:*

$$\forall \sigma \in S_m, x \in \{0,1\}^n : U_O^{\left(U_1^c, U_2^c, \cdots, U_m^c\right)} \cdot |\sigma\rangle|x\rangle$$

$$|\sigma\rangle \left(\prod_{i \in [m]} U_{\sigma^{-1}(i)} \right) \cdot |x\rangle \ .$$

As a brief explanation, the above fact follows by conditioning the execution of unitaries at each step, going (from left to write) on the elements of σ, and the unitary U_i executes if and only if the current element of σ is i. Finally, we put the above pieces together in order to simulate an execution of an OI oracle, using a regular quantum computer. The simulation of the OI oracle is without the amplification part.

Lemma 4.2. *There is a quantum polynomial time (in its input size) algorithm Q, that for input $\{U_i\}_{i \in [m]}$ a set of m unitary circuits, such that all of them act on the same number of qubits n, and $|\psi\rangle$ an n-qubit state, outputs the normalization of the OI state $\mathbf{OI}\left(\{U_i\}_{i \in [m]}, |\psi\rangle\right)$ with probability at least*

$$\left(\frac{\left\| \mathbf{OI}\left(\{U_i\}_{i \in [m]}, |\psi\rangle\right) \right\|}{m!} \right)^2 \ .$$

Proof. The algorithm Q is as follows. First, take the unitary U_R from Claim 4.2 and execute it on $|0^{m \cdot \lceil \log(m) \rceil}\rangle$, to obtain

$$\sum_{i_m \in [m], i_{m-1} \in [m-1], \cdots, i_2 \in [2]} \frac{1}{\sqrt{m!}} |i_m, i_{m-1}, \cdots, i_2\rangle \ .$$

Next, apply the unitary U_{FY} from Claim 4.3 on the above state concatenated with $|0^{m \cdot \lceil \log(m) \rceil}\rangle$, to obtain

$$\sum_{i_m \in [m], i_{m-1} \in [m-1], \cdots, i_2 \in [2]} \frac{1}{\sqrt{m!}} |i_m, i_{m-1}, \cdots, i_2\rangle |\sigma_{(i_m, i_{m-1}, \cdots, i_2)}\rangle$$

$$= \sum_{r \in R_m} \frac{1}{\sqrt{m!}} |r\rangle |\sigma_r\rangle = \frac{1}{\sqrt{m!}} \sum_{\sigma \in S_m} |r_\sigma\rangle |\sigma\rangle ,$$

where the first equality above follows because the function B_{FY} from Claim 4.3 is a bijection, so a sum over R_m can be switched with a sum over S_m.

Next, we execute the unitary U_O (from Fact 4.1) on the right register (containing σ) of the above state, concatenated with the input register containing the state $|\psi\rangle$, where the oracle access of the unitary U_O comes from the fact that the algorithm Q has the unitaries $\{U_i\}_{i \in [m]}$ as part of its input. We get the state

$$\frac{1}{\sqrt{m!}} \sum_{\sigma \in S_m} |r_\sigma\rangle |\sigma\rangle \left(\prod_{i \in [m]} U_{\sigma^{-1}(i)} \right) \cdot |\psi\rangle .$$

We now un-compute in two steps. In the first step we execute U_{FY} once again, to uncompute the information of the permutation, and leave only the information of the element in R_m. After executing U_{FY} and tracing out the middle register that contained σ, the state is

$$\frac{1}{\sqrt{m!}} \sum_{\sigma \in S_m} |r_\sigma\rangle \left(\prod_{i \in [m]} U_{\sigma^{-1}(i)} \right) \cdot |\psi\rangle .$$

Finally, the only probabilistic step of the algorithm is forgetting the information in the left register, containing the information $r \in R_m$. This is done by executing the projective measurement that tries to project the left register on $|0^{m \cdot \lceil \log(m) \rceil}\rangle$. Formally, we execute the unitary U_R once again on the left register. Let us calculate what is the result of the execution.

$$\sum_{\sigma \in S_m} \frac{1}{\sqrt{m!}} (U_R \cdot |r_\sigma\rangle) \left(\prod_{i \in [m]} U_{\sigma^{-1}(i)} \right) \cdot |\psi\rangle$$

$$= \sum_{\sigma \in S_m} \frac{1}{\sqrt{m!}} \left(\sum_{r \in R_m} \frac{1}{\sqrt{m!}} \left(\prod_{j \in [m]} \omega_j^{(r_\sigma)_j \cdot (r)_j} \right) |r\rangle \right) \left(\prod_{i \in [m]} U_{\sigma^{-1}(i)} \right) \cdot |\psi\rangle$$

$$= \frac{1}{m!} \sum_{r \in R_m} |r\rangle \left(\sum_{\sigma \in S_m} \left(\prod_{j \in [m]} \omega_j^{(r_\sigma)_j \cdot (r)_j} \right) \left(\prod_{i \in [m]} U_{\sigma^{-1}(i)} \right) \right) \cdot |\psi\rangle$$

$$= \frac{1}{m!} \sum_{r \in R_m \setminus \{0_{R_m}\}} |r\rangle \left(\sum_{\sigma \in S_m} \left(\prod_{j \in [m]} \omega_j^{(r_\sigma)_j \cdot (r)_j} \right) \left(\prod_{i \in [m]} U_{\sigma^{-1}(i)} \right) \right) \cdot |\psi\rangle$$

$$+ \frac{1}{m!} |0^{m \cdot \lceil \log(m) \rceil}\rangle \left(\sum_{\sigma \in S_m} \left(\prod_{i \in [m]} U_{\sigma^{-1}(i)} \right) \right) \cdot |\psi\rangle .$$

Now, note that the last state in the above summation is

$$\frac{1}{m!} |0^{m \cdot \lceil \log(m) \rceil}\rangle \left(\sum_{\sigma \in S_m} \left(\prod_{i \in [m]} U_{\sigma^{-1}(i)} \right) \right) \cdot |\psi\rangle$$

$$= \frac{\left\|\mathbf{OI}\left(\{U_i\}_{i\in[m]}, |\psi\rangle\right)\right\|}{m!} |0^{m\cdot\lceil\log(m)\rceil}\rangle \frac{1}{\left\|\mathbf{OI}\left(\{U_i\}_{i\in[m]}, |\psi\rangle\right)\right\|} \mathbf{OI}\left(\{U_i\}_{i\in[m]}, |\psi\rangle\right) \quad,$$

which means that the probability to measure the left register to be $|0^{m\cdot\lceil\log(m)\rceil}\rangle$ and obtain the normalized OI state in the right target register is exactly $\left(\frac{\left\|\mathbf{OI}(\{U_i\}_{i\in[m]}, |\psi\rangle)\right\|}{m!}\right)^2$, as needed. $\qquad\square$

To conclude the comparison between quantum computation with and without computable OI, it seems that the only difference is the ability of the OI oracle to amplify the success probability of OI transformations - while a single call to the OI oracle can be simulated efficiently by a quantum computer, for $\ell \in \mathbb{N}$ queries, this complexity increases exponentially. This relation, between quantum computation with and without computable OI is somewhat reminiscent of the relationship between quantum and classical computation.

Specifically, an ℓ-qubit quantum circuit can be broken down (without the loss of generality) into 2-qubit quantum gates. A classical computer can efficiently simulate the action of one step of a quantum computer, which is a single 2-qubit unitary. However, the more gates the classical computer needs to simulate, the complexity of simulation increases exponentially with the number of gates and qubits ℓ.

4.3 Sequentially Invertible Functions and Statistical Difference

As part of this work we define numerous new notions, with no dependence on any non-classical computing or information. Specifically, we define sequentially invertible functions, and a new computational problem, called the Sequentially Invertible Statistical Difference (SISD) problem. We start with the definitions of sequential invertibility.

Definition 4.6 ((r, t, ℓ)-invertible circuit sequence). *For $r, t, \ell \in \mathbb{N}$, an (r, t, ℓ)-invertible circuit sequence, is a sequence of ℓ pairs of randomized circuits $(C_{i,\rightarrow}, C_{i,\leftarrow})_{i\in[\ell]}$ such that for every $i \in [\ell], Y \in \{\rightarrow, \leftarrow\}$, the circuit $C_{i,Y}$ is of size $\leq t$, and such that,*

- *There exists $\bar{k} \in \mathbb{N}$ such that all circuits have input and output size $\bar{k}$, and randomness bounded by r. Additionally, each pair of opposite-direction circuits use the same amount of randomness. Formally, for every $i \in [\ell]$ there exists $r_i \leq r$ such that*

$$\forall Y \in \{\rightarrow, \leftarrow\} : C_{i,Y} : \{0,1\}^{\bar{k}} \times \{0,1\}^{r_i} \to \{0,1\}^{\bar{k}} \quad.$$

- *The circuit directions are inverses of each other, per hard-wired randomness, that is:*

$$\forall i \in [\ell], z \in \{0,1\}^{r_i}, x \in \{0,1\}^{\bar{k}} : x = C_{i,\leftarrow}\left(C_{i,\rightarrow}\left(x;z\right);z\right) \quad.$$

For $r \in \mathbb{N}$, an r-invertible circuit sequence, is an (r, t, ℓ)-invertible circuit sequence, for finite and but unbounded t, ℓ.

Definition 4.7 (Invertible circuit sequence sampling from f). *Let $r, t, \ell \in \mathbb{N}$, $\varepsilon \in [0, 1]$, let $S := (C_{i,\rightarrow}, C_{i,\leftarrow})_{i\in[\ell]}$ an (r, t, ℓ)-invertible circuit sequence with input-output size $\bar{k}$, and let $f : \{0,1\}^{k'} \to \{0,1\}^k$ a function.*

We say that S samples ε-close to f if, $\bar{k} = k$, and for the following distributions D_0, D_1 we have $\|D_0 - D_1\|_{TV} \leq \varepsilon$,

$$D_0 = \left\{ f(z) \mid z \leftarrow \{0,1\}^{k'} \right\}$$

$$D_1 = \left\{ C_{\ell,\rightarrow}\left(\cdots C_{2,\rightarrow}\left(C_{1,\rightarrow}\left(0^{\bar{k}}; z_1\right); z_2\right)\cdots ; z_\ell\right) \;\Big|\; z_1 \leftarrow \{0,1\}^{r_1}, \cdots, z_\ell \leftarrow \{0,1\}^{r_\ell} \right\} \quad.$$

If $\varepsilon = 0$, we simply say that S samples from f.

Definition 4.8 ($(r, t, \ell, \varepsilon)$-sequentially invertible function). *Let $f : \{0, 1\}^{k'} \to \{0, 1\}^{k}$ a function, let $r, t, \ell \in \mathbb{N}$ and let $\varepsilon \in [0, 1]$. We say that f is $(r, t, \ell, \varepsilon)$-sequentially invertible if there exists an (r, t, ℓ)-invertible circuit sequence S that samples ε-close to f.*

Definition 4.9 ($(r(n), t(n), \ell(n), \varepsilon(n))$-sequentially invertible function family). *Let $f = \{f_n\}_{n \in \mathbb{N}}$ a family of functions: For every $n \in \mathbb{N}$, $f_n : \{0, 1\}^{k'(n)} \to \{0, 1\}^{k(n)}$. Let $r, t, \ell : \mathbb{N} \to \mathbb{N}$ and let $\varepsilon : \mathbb{N} \to [0, 1]$. We say that the family of functions f is $(r(n), t(n), \ell(n), \varepsilon(n))$-sequentially invertible if for every $n \in \mathbb{N}$, there exists an $(r(n), t(n), \ell(n))$-invertible circuit sequence S_n that samples $\varepsilon(n)$-close to f_n.*

Note that when considering sequential invertibility, we can interchange between considering the sequential invertibility of functions and circuits.

Remark 4.2 (Interchanging between sequentially invertible circuits and functions). While we define sequential invertibility for functions $f : \{0, 1\}^{k'} \to \{0, 1\}^{k}$, we sometimes refer to sequential invertibility for circuits $C : \{0, 1\}^{k'} \to \{0, 1\}^{k}$. This extension is natural, as circuits are in particular functions – a circuit contains not only the information of the function it computes, but a way to compute the function, using the standard set of classical gates $\{NOT, OR, AND\}$.

The Sequentially Invertible Statistical Difference Problem $\mathsf{SISD}_{a(n), b(n), r(n)}$. Our last new definition in this work is for a new computational problem. We combine the notions of sequential invertibility presented here above, with the computational problem of Statistical Difference SD, to define the Sequentially Invertible Statistical Difference (SISD) Problem. We take the same steps as we took when we defined the problem SD.

Definition 4.10 (The Sequentially Invertible Statistical Difference Problem). *Let $a, b : \mathbb{N} \to [0, 1]$, $r : \mathbb{N} \to \mathbb{N} \cup \{0\}$ functions, such that for every $n \in \mathbb{N}$, $a(n) \leq b(n)$ and $r(n) \in \{0, 1, \cdots, n\}$. The $(a(n), b(n), r(n))$-gap Sequentially Invertible Statistical Difference problem, denoted $\mathsf{SISD}_{a(n), b(n), r(n)}$ is a promise problem, where the input is a pair of sequences,*

$$\left(\left(C_{i, \to}^0, C_{i, \leftarrow}^0 \right)_{i \in [\ell]}, \left(C_{i, \to}^1, C_{i, \leftarrow}^1 \right)_{i \in [\ell]} \right) \ ,$$

where for each $b \in \{0, 1\}$, $\left(C_{i, \to}^b, C_{i, \leftarrow}^b \right)_{i \in [\ell]}$ is an $r(n)$-invertible circuit sequence, where n is the input size,

$$n := \left| \left(\left(C_{i, \to}^0, C_{i, \leftarrow}^0 \right)_{i \in [\ell]}, \left(C_{i, \to}^1, C_{i, \leftarrow}^1 \right)_{i \in [\ell]} \right) \right| \ ,$$

and such that both sequences have the same input and output size k, that is, for every $i \in [\ell]$, $b \in \{0, 1\}$, $Y \in \{\to, \leftarrow\}$,

$$C_{i, Y}^b : \{0, 1\}^k \times \{0, 1\}^{r_i^b(n)} \to \{0, 1\}^k \ .$$

For a pair of circuit sequences, we define their output distributions:

$$D_b := \left\{ C_{\ell, \to}^b \left(\cdots C_{2, \to}^b \left(C_{1, \to}^b \left(0^{k(n)}; z_1 \right); z_2 \right) \cdots ; z_\ell \right) \right\} \ .$$

The problem is defined as follows.

- $\prod_{YES}$ *is the set of pairs of sequences such that $\|D_0 - D_1\|_{TV} \leq a(n)$,*

- $\prod_{NO}$ *is the set of pairs of sequences such that $\|D_0 - D_1\|_{TV} > b(n)$,*

Note that in the above definition, while we consider sequentially invertible circuit sequences as the input to the problem, we focus only on r-invertibility, and ignore the parameters $t \in \mathbb{N}$, $\ell \in \mathbb{N}$, $\varepsilon \in [0, 1]$ from the original definition. The reason is that all parameters, except r, are implicit in the problem input and problem definition. More precisely, the parameters t, ℓ are meaningless as the problem input is the circuit sequence (which defines t, ℓ), and the total variation distance parameter ε is ignored, as there are different total variation distance demands from the circuit sequences.

Finally, as in the case of the (standard) Statistical Difference problem SD, we define the family of problems where the gap is not too small, and formally, lower-bounded inverse-polynomially. To this end we define the family of problems $\mathsf{SISD}_{\mathsf{poly}}$ as follows.

Definition 4.11 (The Polynomial-Gap Sequentially Invertible Statistical Difference Problem). *Let r : $\mathbb{N} \to \mathbb{N} \cup \{0\}$ a function such that for every $n \in \mathbb{N}$, $r(n) \in \{0, 1, \cdots, n\}$. The polynomially-bounded sequentially invertible statistical difference problem, denoted $\mathsf{SISD}_{\mathsf{poly}, r(n)}$ is a set of promise problems. It is defined as follows.*

$$\mathsf{SISD}_{\mathsf{poly}, r(n)} := \bigcup_{\substack{a(n), b(n) : \mathbb{N} \to [0,1], \\ \exists \text{ a polynomial } \mathsf{poly}: \mathbb{N} \to \mathbb{N} \text{ such that } \forall n \in \mathbb{N} : b(n)^2 - 2a(n) + a(n)^2 \geq \frac{1}{\mathsf{poly}(n)}}} \{\mathsf{SISD}_{a(n), b(n), r(n)}\} \ .$$

5 $\mathsf{SISD}_{\mathsf{poly}, O(\log(n))} \in \mathbf{BQP^{OI}}$

In this section we prove that a quantum computer with access to an order interference (OI) oracle is able to solve $\mathsf{SISD}_{\mathsf{poly}, O(\log(n))}$ in polynomial time. For the result in this section, a uniform computable OI oracle (Definition 4.2) suffices.

We first prove a general lemma about an ability of an OI oracle. Basically, the lemma says that an OI oracle can allow an interference between unitary execution choices and not only between unitary execution orders.

Definition 5.1 (Choice Interference State). *Let $\{U_i\}_{i \in [m]}$ a set of $m \in \mathbb{N}$ unitary circuits, all operating on the same number of qubits $n \in \mathbb{N}$. Let $|\psi\rangle$ an n-qubit quantum state. The choice interference state of $\left(\{U_i\}_{i \in [m]}, |\psi\rangle\right)$ is denoted $\mathbf{OI}_c\left(\{U_i\}_{i \in [m]}, |\psi\rangle\right)$ and defined to be*

$$\mathbf{OI}_c\left(\{U_i\}_{i \in [m]}, |\psi\rangle\right) := \sum_{i \in [m]} U_i \cdot |\psi\rangle \ .$$

We are using the following standard fact from quantum computing.

Fact 5.1. *For every $c, N \in \mathbb{N}$ there exists an $\lceil \log(N) \rceil$-qubit quantum circuit $U_{(+c, N)}$ that executes in polynomial-time (in its input size) and possibly uses additional ancillary qubits, such that,*

$$\forall a \in \{0, 1, 2, \cdots, N-1\} : U_{(+c, N)} \cdot |a\rangle = |a + c \mod N\rangle \ .$$

Next, we claim that (a factorial scaling of) a choice interference state can be expressed with an order interference state.

Lemma 5.1 (Norm of simulated choice interference). *Let $\{U_i\}_{i \in [m]}$ a set of $m \in \mathbb{N}$ unitary circuits, all operating on the same number of qubits $n \in \mathbb{N}$. Let $|\psi\rangle$ an n-qubit quantum state.*

For every $i \in [m]$ define the unitary circuit $\tilde{U}_i$ that acts on $n + \lceil \log(m) \rceil$ qubits, and is defined as follows:

- *Execute the circuit U_i^c, which executes U_i on the left n qubits, conditioned on the state of the $\lceil \log(m) \rceil$ qubits on the right being $|0^{\lceil \log(m) \rceil}\rangle$.*

- *Execute the unitary $U_{(+1,2^{\lceil \log(m) \rceil})}$ on the right $\lceil \log(m) \rceil$ qubits.*

Then,

$$\textbf{\textit{OI}} \left(\{\tilde{U}_i\}_{i \in [m]}, |\psi\rangle |0^{\lceil \log(m) \rceil}\rangle \right) = (m-1)! \cdot \left(\textbf{\textit{OI}}_c \left(\{U_i\}_{i \in [m]}, |\psi\rangle \right) \otimes |m \bmod 2^{\lceil \log(m) \rceil}\rangle \right) \ .$$

Proof. Let $\sigma \in S_m$ an execution order of the unitaries in $\{\tilde{U}_i\}_{i \in [m]}$ on the state $|\psi\rangle |0^{\lceil \log(m) \rceil}\rangle$. We consider what happens to the state in each of the unitary execution orders $\sigma \in S_m$. Recall that executing the unitary $\tilde{U}_i$ means executing U_i^c and then executing $U_{(+1,2^{\lceil \log(m) \rceil})}$. Two things can be easily verified.

- For $i \in [m]$ and any execution order $\sigma \in S_m$ such that $\sigma^{-1}(i) = 1$, the state in the left n qubits at the end of the computation is $U_i \cdot |\psi\rangle$. In simple terms, the only unitary that effectively executes on the side of the input state $|\psi\rangle$ is the first one in the ordering.

- When we look at the state of the right $\lceil \log(m) \rceil$-qubit counter register, for *any* permutation $\sigma \in S_m$, the state is always $m \mod 2^{\lceil \log(m) \rceil}$.

Together, this means that the OI state is,

$$\textbf{OI} \left(\{\tilde{U}_i\}_{i \in [m]}, |\psi\rangle |0^{\lceil \log(m) \rceil}\rangle \right) \underset{\text{(as we explained above)}}{=} \sum_{\sigma \in S_m, i \in [m] \text{ such that } \sigma^{-1}(i)=1} U_i \cdot |\psi\rangle |0^{\lceil \log(m) \rceil}\rangle$$

$$= \sum_{i \in [m]} \left(\sum_{\sigma \in S_m \text{ such that } \sigma^{-1}(i)=1} U_i \cdot |\psi\rangle |m \bmod 2^{\lceil \log(m) \rceil}\rangle \right)$$

$$= (m-1)! \sum_{i \in [m]} (U_i \cdot |\psi\rangle) \otimes |m \bmod 2^{\lceil \log(m) \rceil}\rangle$$

$$:= (m-1)! \cdot \left(\textbf{OI}_c \left(\{U_i\}_{i \in [m]}, |\psi\rangle \right) \otimes |m \bmod 2^{\lceil \log(m) \rceil}\rangle \right) \ .$$

$\square$

Finally, the above implies the below corollary lemma, which says that an OI oracle can be used in order to simulate oracle access to choice interference with a factorial reduction in needed complexity, for successfully executing the transformation.

Lemma 5.2 (Choice interference oracle from order interference oracle). *Let $\mathcal{O}_{\textbf{OI}}(\cdot, \cdot, \cdot)$ a uniform order interference oracle as in Definition 4.2. There is a quantum procedure Q_{CI} with oracle access to $\mathcal{O}_{\textbf{OI}}$ and the following input: $\{U_i\}_{i \in [m]}$ a set of $m \in \mathbb{N}$ unitary circuits, all operating on the same number of qubits $n \in \mathbb{N}$, $|\psi\rangle$ an n-qubit quantum state, and $\lambda \in \mathbb{N}$.*

$Q_{CI}\left(\{U_i\}_{i \in [m]}, |\psi\rangle, \lambda\right)$ executes in time $O\left(\lambda + \log(m) \cdot \left(\sum_{i \in [m]} |U_i|\right)\right)$ and with probability at least

$$\frac{\frac{\left\| \textbf{\textit{OI}}_c\left(\{U_i\}_{i \in [m]}, |\psi\rangle\right) \right\|}{m}}{\frac{\left\| \textbf{\textit{OI}}_c\left(\{U_i\}_{i \in [m]}, |\psi\rangle\right) \right\|}{m} + \frac{1}{\lambda}} \ ,$$

outputs the normalization of the choice interference state $\textbf{\textit{OI}}_c\left(\{U_i\}_{i \in [m]}, |\psi\rangle\right)$.

Main Theorems. At this point we are ready to prove the two main theorems of this section. We will use the following standard fact from quantum computing.

Fact 5.2 (Unitary circuits for Classical Bijections). *Let $C_\rightarrow : \{0,1\}^k \rightarrow \{0,1\}^k$, $C_\leftarrow : \{0,1\}^k \rightarrow \{0,1\}^k$ two classical circuits that are inverses of each other, that is,*

$$\forall x \in \{0,1\}^k : C_\leftarrow(C_\rightarrow(x)) = x \ .$$

Then, there exists a k-qubit unitary circuit U_C with complexity $O(|C_\rightarrow| + |C_\leftarrow|)$ such that,

$$\forall x \in \{0,1\}^k : U_C \cdot |x\rangle = |C_\rightarrow(x)\rangle \ .$$

Moreover, there is a classical polynomial-time deterministic Turing Machine M that computes the classical description of the unitary: $(C_\rightarrow, C_\leftarrow) \rightarrow U_C$.

We will use the following claim in the proof.

Claim 5.1 (Minimal norm of choice interference for classical positive operations). *Let $|\psi\rangle = \sum_{x \in \{0,1\}^k} \alpha_x \cdot |x\rangle$ a k-qubit state such that for all $x \in \{0,1\}^k$, the amplitude α_x is real and non-negative. Let $\{U_i\}_{i \in [m]}$ a set of m unitary transformations, all act on the same number of qubits k, and such that all unitaries map from classical to classical states, formally:*

$$\forall i \in [m], x \in \{0,1\}^k \ \exists y_{i,x} \in \{0,1\}^k : U_i \cdot |x\rangle = |y_{i,x}\rangle \ .$$

Then, we have the following lower bound on the norm of the choice interference state

$$\left\| \sum_{i \in [m]} U_i \cdot |\psi\rangle \right\| \geq \sqrt{m} \ .$$

Proof. We calculate the squared norm of the choice interference state for the proof.

$$\left\| \sum_{i \in [m]} U_i \cdot |\psi\rangle \right\|^2 = \left\| \sum_{i \in [m]} \left(U_i \cdot \sum_{x \in \{0,1\}^k} \alpha_x \cdot |x\rangle \right) \right\|^2$$

$$= \left\| \sum_{i \in [m]} \left(\sum_{x \in \{0,1\}^k} \alpha_x \cdot |y_{i,x}\rangle \right) \right\|^2$$

$$= \left\| \sum_{x \in \{0,1\}^k} \left(\alpha_x \sum_{i \in [m]} |y_{i,x}\rangle \right) \right\|^2$$

$$= \left(\sum_{x \in \{0,1\}^k, i \in [m]} \alpha_x \langle y_{i,x}| \right) \cdot \left(\sum_{z \in \{0,1\}^k, j \in [m]} \alpha_z |y_{j,z}\rangle \right)$$

$$= \sum_{x,z \in \{0,1\}^k, i,j \in [m]} \alpha_x \alpha_z \langle y_{i,x}||y_{j,z}\rangle$$

$$\underset{\text{(all summands are non-negative)}}{\geq} \sum_{x \in \{0,1\}^k, i \in [m]} \alpha_x \alpha_x \langle y_{i,x}||y_{i,x}\rangle$$

$$= \sum_{x \in \{0,1\}^k} \alpha_x^2 \cdot m = m \ .$$

$\square$

We next prove the first main theorem of this section.

Theorem 5.1 (Quantum State Generation for Invertible Circuit Sequences). *There exists $Q^{OI}(\cdot, \cdot)$ a quantum algorithm with oracle access to a uniform OI oracle (as in Definition 4.2), that gets as input (1) an amplification parameter $\delta \in \mathbb{N}$ and (2) an r-invertible circuit sequence $C := (C_{i,\rightarrow}, C_{i,\leftarrow})_{i \in [\ell]}$ with output size k of the circuits, and the total description size of the pair of sequences is denoted with n.*

The algorithm Q^{OI} executes in time $O(\delta \cdot \mathsf{poly}(n) \cdot 2^r)$ for some polynomial $\mathsf{poly}(\cdot)$, and outputs a success/fail bit $b \in \{0,1\}$ together with a k-qubit quantum state. The algorithm succeeds (outputs $b = 1$) with probability $\geq 1 - \frac{1}{\delta}$ and along with the bit outputs the (normalization of the) output distribution state of the circuit sequence:

$$|C(R)\rangle := \sum_{\substack{z_1 \in \{0,1\}^{r_1}, \\ \vdots \\ z_\ell \in \{0,1\}^{r_\ell}}} \left| C_{\ell,\rightarrow}\left(\cdots C_{2,\rightarrow}\left(C_{1,\rightarrow}\left(0^k; z_1\right); z_2\right)\cdots; z_\ell\right)\right\rangle . \tag{8}$$

Proof. Let $k \in \mathbb{N}$ the output size of the invertible circuit sequence. For every $i \in [\ell]$, for every $z_i \in \{0,1\}^{r_i}$, since this is an invertible circuit sequence, the two circuits $C_{i,\rightarrow}(\cdot; z_i)$, $C_{i,\leftarrow}(\cdot; z_i)$ are inverses of each other. It follows from Fact 5.2 that we can efficiently compute a description of the unitary U_{i,z_i} that acts on k qubits, has complexity $O(|C_{i,\rightarrow}(\cdot; z_i)| + |C_{i,\leftarrow}(\cdot; z_i)|)$ and maps,

$$\forall x \in \{0,1\}^k : U_{i,z_i} \cdot |x\rangle = |C(x; z_i)\rangle .$$

Next, observe that for every $i \in [\ell]$, for every k-qubit state $|\phi\rangle$ such that all of its amplitudes are real and non-negative, the vector $\sum_{z_i \in \{0,1\}^{r_i}} U_{i,z_i} \cdot |\phi\rangle$ has only real and non-negative amplitudes. This follows because for every $i \in [\ell]$, the unitaries in $\{U_{iz_i}\}_{z_i \in \{0,1\}^{r_i}}$ all map classical to classical states. It follows that the normalization of the choice interference state $\sum_{z_i \in \{0,1\}^{r_i}} U_{i,z_i} \cdot |\phi\rangle$ is a quantum state with only real and non-negative amplitudes.

According to Claim 5.1, for every $i \in [\ell]$ and every such $|\phi\rangle$ as above,

$$\left\| \sum_{z_i \in \{0,1\}^{r_i}} U_i \cdot |\phi\rangle \right\| \geq \sqrt{2^{r_i}} .$$

It thus follows that for every k-qubit state $|\phi\rangle$ such that all of its amplitudes are real and positive, we have,

$$\frac{\left\| \mathbf{OI}_c\left(\{U_{i,z_i}\}_{z_i \in \{0,1\}^{r_i}}, |\phi\rangle\right)\right\|}{2^{r_i}} \geq \frac{1}{\sqrt{2^{r_i}}} \geq \frac{1}{\sqrt{2^r}} . \tag{9}$$

By induction on i, one can verify that for every $i \in [\ell]$, executing the choice interference oracle (the algorithm Q_{CI} from Lemma 5.2) on the triplet $\left(\{U_{i,z_i}\}_{z_i \in \{0,1\}^{r_i}}, |C_{1,\cdots,i-1}(R_{1,\cdots,i-1})\rangle, \delta \cdot \ell \cdot \sqrt{2^r}\right)$, has the following properties.

- The oracle call takes time complexity $O\left(\delta \cdot \ell \cdot \sqrt{2^r} + r_i \cdot \sum_{z_i \in \{0,1\}^{r_i}} |U_{i,z_i}|\right)$, which is bounded by $O\left(\delta \cdot n^2 \cdot 2^r\right)$, because $\ell \leq n$ and $r_i \cdot \sum_{z_i \in \{0,1\}^{r_i}} |U_{i,z_i}| \leq n \cdot 2^{r_i} \leq n \cdot 2^r$.

- The choice interference transformation succeeds with probability $\geq 1 - \frac{1}{\delta \cdot \ell}$, due to Lemma 4.1 together with Equation 9.

- After a successful transformation, the output state is

$$|C_{1,\cdots,i}(R_{1,\cdots,i})\rangle := \sum_{\substack{z_1 \in \{0,1\}^{r_1}, \\ \vdots \\ z_i \in \{0,1\}^{r_i}}} \left| C_{i,\rightarrow}\left(\cdots C_{1,\rightarrow}\left(0^k; z_1\right)\cdots; z_i\right)\right\rangle ,$$

36

which follows based on the equality,

$$\mathbf{OI}_c\left(\{U_{i,z_i}\}_{z_i\in\{0,1\}^{r_i}}, |C_{1,\cdots,i-1}(R_{1,\cdots,i-1})\rangle\right) = |C_{1,\cdots,i}(R_{1,\cdots,i})\rangle \ .$$

It can be verified that if all ℓ executions of the algorithm Q_{CI} succeed, then by the correctness of the induction, the state we get at the end is the desired normalization of $|C(R)\rangle$. Finally, all ℓ calls together take time complexity $\ell \cdot O\left(\delta \cdot n^2 \cdot 2^r\right) = O\left(\delta \cdot n^3 \cdot 2^r\right)$, and the probability that all calls succeed is $1 - p$ where p is the probability that one of the calls fail. As we saw, for each call, the probability to fail is bounded by $\frac{1}{\delta \cdot \ell}$. By union bound, we have $p \leq \ell \cdot \frac{1}{\delta \cdot \ell} = \frac{1}{\delta}$, which finishes the proof. $\qquad\square$

Finally, we conclude with the second main theorem, that says that a quantum computer with computable order interference can be used to solve the sequentially invertible statistical difference problem in quantum polynomial time, whenever the sequential invertibility parameter r is at most logarithmic in the input size.

Theorem 5.2 ($\mathsf{SISD}_{\mathsf{poly},O(\log(n))} \in \mathbf{BQP^{OI}}$). *Let* $\mathsf{SISD}_{a,b,r} := \left(\prod_{YES}, \prod_{NO}\right)$ *an* SISD *promise problem, such that* $\mathsf{SISD}_{a,b,r} \in \mathsf{SISD}_{\mathsf{poly},\,O(\log(n))}$. *Then, there is* $M^{\mathbf{OI}}$ *a QPT algorithm with oracle access to a computable uniform order interference oracle* $\mathcal{O}_{\mathbf{OI}}$ *(Definition 4.2), that for every* $x \in \prod_{YES}$, $M^{\mathbf{OI}}(x) = 1$ *with probability at least* $p(x)$ *and for every* $x \in \prod_{NO}$, $M^{\mathbf{OI}}(x) = 0$ *with probability at least* $p(x)$, *such that*

$$\forall x \in \left(\prod_{YES} \cup \prod_{NO}\right) : p(x) \geq 1 - 2^{-\mathsf{poly}(|x|)} \ ,$$

for some polynomial $\mathsf{poly}(\cdot)$.

Proof. Let the following pair of sequences an input for the $\mathsf{SISD}_{a,b,r}$ problem:

$$\left(\left(C^0_{i,\rightarrow}, C^0_{i,\leftarrow}\right)_{i\in[\ell]}, \left(C^1_{i,\rightarrow}, C^1_{i,\leftarrow}\right)_{i\in[\ell]}\right) \ .$$

Denote by n the description size of the pair of sequences, and since $\mathsf{SISD}_{a,b,r} \in \mathsf{SISD}_{\mathsf{poly},\,O(\log(n))}$ it follows that

- There exists a polynomial $\mathsf{poly} : \mathbb{N} \to \mathbb{N}$ such that for every $n \in \mathbb{N}, b(n)^2 - 2a(n) + a(n)^2 \geq \frac{1}{\mathsf{poly}(n)}$.

- There exists a constant $c \in \mathbb{N}$ such that for every $n \in \mathbb{N}, r(n) \leq c \cdot \log(n)$.

Generating many copies of the output distribution states of the two circuits. Note that for $b \in \{0,1\}$, if we execute the algorithm $Q^{\mathbf{OI}}\left(\left(C^b_{i,\rightarrow}, C^b_{i,\leftarrow}\right)_{i\in[\ell]}, 2\right)$ from Theorem 5.1, the call takes complexity $O\left(2 \cdot \mathsf{poly}'(n) \cdot 2^r\right)$ and succeeds with probability $\geq 1 - \frac{1}{2} = \frac{1}{2}$. If we execute $3n$ tries, then we have at least a single successful state generation, with probability $\geq 1 - 2^{-3n}$. For each $b \in \{0,1\}$ we execute $N := 16 \cdot \mathsf{poly}(n)^2 \cdot 12 \cdot n$ times the previous procedure (where you execute $3n$ tries, and if one of them succeeds, you take the state), and if one of the $2 \cdot N$ attempts fails, the algorithm halts and outputs $\perp$.

If all attempts succeed, then for each $b \in \{0,1\}$, we now have N copies of $|C^b(R)\rangle$. Note that all attempts together take complexity $N \cdot O\left(2 \cdot \mathsf{poly}'(n) \cdot 2^r\right)$ which is $\leq \tilde{\mathsf{poly}}(n)$ for some polynomial $\tilde{\mathsf{poly}}$ (this follows because $r \leq c \cdot \log(n)$ for a constant $c \in \mathbb{N}$, and thus $2^r \leq n^c$ is bounded by a polynomial in n as well). Also, by union bound, all attempts together succeed with probability $\geq 1 - \frac{2 \cdot N}{2^{3n}} \geq 1 - \frac{1}{2^{2n}}$. This step of the algorithm is the only one that uses computable order interference and the OI oracle. The next steps use only standard quantum computation.

Repeated swap tests and decision. Let us list the N copies of each of the quantum states as

$$|C^0(R)\rangle_{Reg_{0,1}}, |C^0(R)\rangle_{Reg_{0,2}}, \cdots, |C^0(R)\rangle_{Reg_{0,N}} \ ,$$

$$|C^1(R)\rangle_{Reg_{1,1}}, |C^1(R)\rangle_{Reg_{1,2}}, \cdots, |C^1(R)\rangle_{Reg_{1,N}} .$$

For each $j \in [N]$, we now execute a swap test (the algorithm Q_{ST} from Theorem 3.1) on the pair of k-qubit states $|C^0(R)\rangle_{Reg_{0,j}}, |C^1(R)\rangle_{Reg_{1,j}}$. For every execution of the swap test the complexity is $O(k) \le O(n)$, and over all N executions the complexity is polynomial in n.

Let $\{b_i\}_{i \in [N]}$ the N results of the swap test executions. We take an average of the results, that is

$$A := \frac{\sum_{i \in [N]} b_i}{N} .$$

The algorithm decides where the input belongs by a threshold: We take $T := 1 - \frac{b(n)^2 + 2 \cdot a(n) - a(n)^2}{4}$, and if $A > T$ we output 1, and otherwise (i.e., $A \le T$), we output 0.

Final analysis. The first thing to verify is that the total execution time of the algorithm is polynomial in n, as both steps are polynomial in n. We next explain why the probability to output the correct answer (i.e., output 1 when the input is in $\prod_{\text{YES}}$ and output 0 when the input is in $\prod_{\text{NO}}$) is exponentially (in the input size n) close to 1.

By the correctness of the swap test, for every $i \in [N]$ of the swap test executions, the probability for $b_i = 1$ is $p := \frac{1 + |\langle C^0(R) || C^1(R) \rangle|^2}{2}$. Observe that the inner product $\langle C^0(R) || C^1(R) \rangle$ equals the fidelity between the output distributions of C^0, C^1, thus $|\langle C^0(R) || C^1(R) \rangle|^2 = F(C^0(R), C^1(R))^2$. Recall the following known relation between total variation distance and fidelity of classical distributions:

$$1 - F(D_0, D_1) \le \|D_0 - D_1\|_{TV} \le \sqrt{1 - F(D_0, D_1)^2} .$$

We have the following conclusions.

- If the input pair of sequences is in $\prod_{\text{YES}}$, then $\|C^0(R) - C^1(R)\|_{TV} \le a(n) \le 1$ and,

$$p = \frac{1 + F(C^0(R), C^1(R))^2}{2} \ge \frac{1 + (1 - \|C^0(R) - C^1(R)\|_{TV})^2}{2} \ge \frac{1 + (1 - a(n))^2}{2}$$

$$= \frac{1 + 1 - 2a(n) + a(n)^2}{2} = 1 - \frac{2a(n) - a(n)^2}{2} .$$

- If the input pair of sequences is in $\prod_{\text{NO}}$, then $\|C^0(R), C^1(R)\|_{TV} > b(n) \ge 0$ and,

$$p = \frac{1 + F(C^0(R), C^1(R))^2}{2} \le \frac{1 + 1 - \|C^0(R) - C^1(R)\|_{TV}^2}{2} < \frac{1 + 1 - b(n)^2}{2}$$

$$= 1 - \frac{b(n)^2}{2} .$$

Next, for each of the N executions of the swap test, note we have a binary random variable b_i that is 1 with probability $p = \frac{1 + |\langle C^0(R) || C^1(R) \rangle|^2}{2}$. It follows that the expectation of the average of executions of the swap test (that is, the expectation of A) is the same, p. By Chernoff's bound 3.2 it follows that for every $\varepsilon \in (0, 1)$,

$$\Pr_{b_1, \cdots, b_N} \left[|A - p| \ge \varepsilon \cdot p \right] \le 2 \cdot e^{-N \cdot \frac{p \cdot \varepsilon^2}{3}} .$$

Note that the threshold $T := 1 - \frac{b(n)^2 + 2 \cdot a(n) - a(n)^2}{4}$ is exactly the average between two numbers: One is the lower bound $B_{\prod_{\text{YES}}} := 1 - \frac{2a(n) - a(n)^2}{2}$ on p in case the input is in $\prod_{\text{YES}}$, and the second number is the upper bound $B_{\prod_{\text{NO}}} := 1 - \frac{b(n)^2}{2}$ on p in case the input is in $\prod_{\text{NO}}$. This means that T sits on the real

38

number line exactly in the middle between $B_{\prod_{\text{YES}}}$ and $B_{\prod_{\text{NO}}}$. In order to complete the proof, it will be sufficient to (1) understand a lower bound L on the distance $|B_{\prod_{\text{YES}}} - B_{\prod_{\text{NO}}}|$ (half of this lower bound acts as a lower bound for $|B_{\prod_{\text{YES}}} - T|$ and $|B_{\prod_{\text{NO}}} - T|$), and (2) show that the probability for A to deviate from the average p by more than $L/2$ is exponentially small, in each of the cases.

The first part is easy: We know

$$|B_{\prod_{\text{YES}}} - B_{\prod_{\text{NO}}}| = 1 - \frac{2a(n) - a(n)^2}{2} - 1 + \frac{b(n)^2}{2} = \frac{b(n)^2 - 2a(n) + a(n)^2}{2} \geq \frac{1}{2 \cdot \text{poly}(n)} \ .$$

This means that by taking half of the above, which is $\frac{1}{4 \cdot \text{poly}(n)}$, we have a lower bound from both $|B_{\prod_{\text{YES}}} - T|$ and $|B_{\prod_{\text{NO}}} - T|$.

Finally, by the above stated instance of the Chernoff bound, for $\varepsilon := \frac{1}{4 \cdot \text{poly}(n)}$, the probability for A to deviate from p with an amount $\varepsilon \cdot p \leq \varepsilon = \frac{1}{4 \cdot \text{poly}(n)}$, is bounded by

$$2 \cdot e^{-N \cdot \frac{p \cdot \varepsilon^2}{3}} \underset{(p \geq \frac{1}{2})}{\leq} 2 \cdot e^{-N \cdot \frac{\varepsilon^2}{6}} = 2 \cdot e^{-\frac{N}{6 \cdot 16 \cdot \text{poly}(n)^2}}$$

$$= 2 \cdot e^{-\frac{16 \cdot \text{poly}(n)^2 \cdot 12 \cdot n}{6 \cdot 16 \cdot \text{poly}(n)^2}} = 2 \cdot e^{-2n} \ .$$

To conclude, recall that the probability for the first part of the algorithm to succeed is at least $1 - 2^{-2n}$, and the probability for the second part of the algorithm to be correct, conditioned on the first part being successful, is at least $1 - 2 \cdot e^{-2n}$. Overall, if both parts succeed, which happens with probability at least $(1 - 2^{-2n}) \cdot (1 - 2 \cdot e^{-2n}) \geq 1 - 2^{-n} = 1 - 2^{-|x|}$, the algorithm outputs the correct answer for the input, as desired. $\qquad\square$

6 $\ \ \mathsf{GI} \leq_p \mathsf{SISD}_{\text{poly}, O(\log(n))}$

In this section we prove our main theorem regarding the computational problem of Graph Isomorphism GI.

Theorem 6.1 (Reduction from GI to $\mathsf{SISD}_{\frac{n}{2n}, 1-\frac{n}{2n}, \log(n)}$)**.** *There exists a classical deterministic polynomial time Turing machine M that computes the reduction,*

$$\mathsf{GI} \leq_p \mathsf{SISD}_{a(n), b(n), r(n)} \ ,$$

for $a(n) := \frac{n}{2n}$, $b(n) := 1 - \frac{n}{2n}$, $r(n) := \log(n)$.

Note that for the values of $a(n), b(n)$ above we have $b(n)^2 - 2a(n) + a(n)^2$ is at least inverse polynomially large, and also $r(n) = \log(n) = O(\log(n))$, and thus $\mathsf{SISD}_{a(n), b(n), r(n)} \in \mathsf{SISD}_{\text{poly}, O(\log(n))}$. We proceed to the proof of the theorem.

Proof. Let (G_0, G_1) an input for the GI problem, and let $n \in \mathbb{N}$ denote the number of vertices in each of the graphs, and assume that the input graphs are represented by adjacency matrices. So, the description size of the input is

$$|(G_0, G_1)| = 2 \cdot n^2 \ .$$

The reduction executes in classical deterministic polynomial time and will output a pair (C^0, C^1) of invertible circuit sequences (as in Definition 4.6). Let $\ell := (n-1) \cdot n$.

Definition of the invertible circuit sequence C and then the pair of sequences (C^0, C^1). We define the invertible circuit sequence $C := (C_{i, \rightarrow}, C_{i, \leftarrow})_{i \in [\ell]}$ below, where all ℓ circuit pairs have input/output size n^2 (which we think of as an adjacency matrix for a graph with n vertices) and randomness of exactly $\log(n)$ bits (formally, we use $\lceil \log(n) \rceil$ bits). We will later define the circuits C^0, C^1, as a function of the circuit sequence C.

1. For $i = n, n-1, \cdots, 3, 2$ (that is, starting from $i = n$ and moving down to $i = 2$), we define,

    (a) For $j \in [n]$, the circuit $C_{(i,j),\rightarrow}$ acts as follows. The circuit's main input is a graph $G = ([n], E_G)$ (in the data structure of adjacency matrix of n^2 bits) and auxiliary input is the randomness $z_{i,j} \in \{0,1\}^{\lceil \log(n) \rceil}$.

        • Define $d_i := \lceil \log(i) \rceil$, and the circuit computes only on the first d_i bits of randomness in $z_{i,j}$, and ignores the rest. Denote by $z'_{i,j} \in \{0,1\}^{d_i}$ the first d_i digits of the randomness.
        • The circuit interprets the string $z'_{i,j} \in \{0,1\}^{d_i}$ as a uniformly random number in $[2^{d_i}]$ (by adding 1 to the number derived from the binary representation of the string), denote this number by $k_{i,j} \in [2^{d_i}]$. In case $k_{i,j} > i$ the circuit does nothing and its function is the identity.
        • In case $k_{i,j} \le i$ the circuit $C_{(i,j),\rightarrow}$ computes the graph $\sigma_{(i,k_{i,j})}(G)$ as its output, where $\sigma_{(i,k_{i,j})} \in S_i$ is the permutation on the elements $[i]$, that acts trivially on all elements that are in $[i] \setminus \{i, k_{i,j}\}$, and swaps between i and $k_{i,j}$.

We defined what are the forward computations, and the definition of the inverse circuits follows: For every $i \in \{n, n-1, \cdots, 3, 2\}$, $j \in [n]$, the circuit $C_{(i,j),\leftarrow}$ is exactly identical to the circuit $C_{(i,j),\rightarrow}$. It can be verified by the reader that indeed for every $i \in \{n, n-1, \cdots, 3, 2\}$, $j \in [n]$, the circuits $C_{(i,j),\rightarrow}, C_{(i,j),\leftarrow}$ are inverses of each other, because making the same swap twice leads to the identity permutation.

Regarding the output of the reduction, which should be a pair of invertible circuits sequences (C^0, C^1): For $b \in \{0,1\}$, the sequence C^b is the same as the above sequence C, with one adding: In the beginning, there is a deterministic (randomness-free) circuit $C^b_{0,\rightarrow}$ that adds the description of the graph G_b, by simply adding the bits (modulo 2, also known as XOR-ing) of the adjacency matrix of G_b. Accordingly, the inverse $C^b_{0,\leftarrow}$ of that first circuit $C^b_{0,\rightarrow}$, is subtracting the description of G_b (which makes the circuits equal, because subtracting and adding modulo 2 is identical). To conclude, it can be easily verified that both C^0 and C^1 are legal $(\log(n))$-invertible circuit sequences.

Properties of the output distributions of the circuits C^0, C^1. For $b \in \{0,1\}$, one can observe that the output distribution of C^b (when starting from a string of n^2 zeros) is as follows.

   • For every $i \in \{n, n-1, \cdots, 3, 2\}$, if for all $j \in [n]$, the randomness $z_{i,j} \in \{0,1\}^{\log(n)}$ was such that the number $k_{i,j}$ (which is derived from $z'_{i,j} \in \{0,1\}^{d_i}$, the first $d_i := \lceil \log(i) \rceil$ bits of $z_{i,j}$) is $> i$, then the graph stays the same after passing the entire outer iteration i. However, note that $2^{d_i} \le 2 \cdot i$, which means that a uniformly random $k_{i,j} \in [2^{d_i}]$ is bounded by i with probability at least $\frac{1}{2}$. Thus, the probability that for all $j \in [n]$, the sample $k_{i,j} > i$, is bounded by 2^{-n}. By union bound, the probability that there exists an index $i \in \{n, n-1, \cdots, 3, 2\}$ such that for every $j \in [n]$, $k_{i,j}$ caused no change, is bounded by $\frac{n}{2^n}$.

   • In the other case, that happens with probability $\ge 1 - \frac{n}{2^n}$, for every $i \in \{n, n-1, \cdots, 3, 2\}$, there is at least a single try $j \in [n]$ such that $k_{i,j} \le i$. Observe that in this second case the output distribution of C^b is as follows. For an iteratively decreasing $i = n, n-1, \cdots, 3, 2$, apply $\sigma_{(i,r_i)}$ to the graph, for a uniformly random $r_i \in [i]$. The above distribution is exactly the description of the Fisher-Yates algorithm, and thus by Fact 3.2, generates a uniformly random permutation $\sigma \in S_n$, and applies it to the input graph G_b.

The above means that the output distribution of C^b is statistically close (has total variation distance bounded by $\frac{n}{2^n}$) to $\sigma(G_b)$, for a uniformly random permutation $\sigma \in S_n$.

Soundness and completeness of the reduction. The next step of our proof is to show that the reduction is correct, that is, that when $(G_0, G_1) \in \prod_{\text{YES}}$ then the output distributions of the circuits, denoted

$\mathcal{D}\left(C^0\right)$, $\mathcal{D}\left(C^1\right)$ respectively, are statistically close, and when the input is in $\prod_{\text{NO}}$, then the output distributions are statistically far.

Both soundness and completeness follow rather easily at this point. For soundness, in case $(G_0, G_1) \in \prod_{\text{NO}}$, it means the graphs are non-isomorphic, which, by Fact 3.1, means that the supports of the distributions $\sigma\left(G_0\right)$, $\sigma\left(G_1\right)$ are disjoint, and thus $\left\|C^0 - C^1\right\|_{TV} \geq 1 - \frac{n}{2^n}$. For completeness, in case $(G_0, G_1) \in \prod_{\text{YES}}$, it means the graphs are isomorphic, which, by Fact 3.1, means that the distributions $\sigma\left(G_0\right)$, $\sigma\left(G_1\right)$ are identical, and thus $\left\|C^0 - C^1\right\|_{TV} \leq \frac{n}{2^n}$. $\qquad\square$

7 $\mathsf{GapCVP}_{O(n^2)} \leq_p \mathsf{SISD}_{\mathsf{poly}, O(\log(n))}$

In this section we prove our main theorem regarding GapCVP.

Theorem 7.1 (Reduction from $\mathsf{GapCVP}_{O(n^2)}$ to $\mathsf{SISD}_{\frac{1}{2}+\frac{1}{2^n}, 1, 1}$)**.** *There exists a positive absolute constant $c \in \mathbb{R}_{>0}$ and a classical deterministic polynomial time Turing machine M that computes the reduction,*

$$\mathsf{GapCVP}_{c \cdot n^2} \leq_p \mathsf{SISD}_{a(n), b(n), r(n)} \ ,$$

for $a(n) := \frac{1}{2} + \frac{1}{2^n}$, $b(n) := 1$, $r(n) := 1$.

Note that for the values of $a(n)$, $b(n)$ above we have $b(n)^2 - 2a(n) + a(n)^2$ is at least inverse polynomially large, and also $r(n) = 1 = O(\log(n))$, and thus $\mathsf{SISD}_{a(n), b(n), r(n)} \in \mathsf{SISD}_{\mathsf{poly}, O(\log(n))}$. We proceed to the proof of the theorem.

Proof. Let $(\mathbf{B}, \mathbf{t}, d)$ an input for the $\mathsf{GapCVP}_{g(n)}$ problem, for $g(n) = c_g \cdot n^2$ for $c_g := 256 \cdot \frac{c_0}{c_1} \cdot \pi$, where $c_0, c_1 \in \mathbb{R}_{>0}$ are two positive absolute constants, specified later, and $n \in \mathbb{N}$ is the dimension of the lattice basis $\mathbf{B}$ and the vector $\mathbf{t}$, and $b \in \mathbb{N}$ is the size of the binary representation of the largest (in absolute value) number in $\mathbf{B}$ and the vector $\mathbf{t}$. So, the description size of the input is

$$|(\mathbf{B}, \mathbf{t}, d)| = n^2 \cdot b + n \cdot b + \log_2(d) \ .$$

The reduction executes in classical deterministic polynomial time and will output a pair $\left(C^0, C^1\right)$ of invertible circuit sequences (as in Definition 4.6). Let $\ell := \ell_{\mathcal{L}} + 1 + \ell_G + 1$, where $\ell_{\mathcal{L}} := (m+1) \cdot n$ and $\ell_G := \kappa \cdot (\beta + 1) \cdot n$, such that,

- $m := \lceil \log_2(M) \rceil$ for $M := 2^n \cdot \left(\sum_{i \in [n]} \|\mathbf{b}_i\| + \|\mathbf{t}\| + d\right)$,

- $\kappa := c_\kappa \cdot n^2$ for $c_\kappa := c_0^2 \cdot 64$ (as in the case of c_g, the constant c_0 will be specified later).

- $\beta := \lfloor \log_2(B) \rfloor$, for $B := \frac{g(n) \cdot d}{2 \cdot \kappa \cdot \sqrt{n}}$.

Let $s := \lceil b + m + \log(n \cdot g(n) \cdot d) \rceil$ and note that a binary string of size $s+1$ can be used to represent any integer of absolute value bounded by

$$2^s - 1 \geq 2^b \cdot M \cdot n \cdot g(n) \cdot d - 1 \ ,$$

by using the first bit as a decider for the sign of the number, and the rest of the s bits as the binary representation of the absolute value of the number (the representation is not necessarily bijective, of course). In the following proof we thus use binary strings of size $s+1$ to represent numbers in the set $\mathbb{Z}_{(-(2^s-1), 2^s-1)}$.

Definition of the circuit sequence C and the pair $\left(C^0, C^1\right)$. We define the invertible circuit sequence $C := (C_{i, \rightarrow}, C_{i, \leftarrow})_{i \in [\ell]}$ below, where all ℓ circuit pairs have input/output size $n \cdot (s+1)$ (which we think of as an integer vector $\mathbf{v} \in \mathbb{Z}_{(-(2^s-1), 2^s-1)}^n$) and randomness bounded by 1 (that is, each circuit either

uses a single bit of randomness or it is deterministic, and does not use randomness at all). In the below circuits, all arithmetic (which comes down to additions and multiplications) is modulo in absolute value of $2^s - 1$.

1. *Constructing a random lattice vector with uniform coordinates vector in $\mathbb{Z}^n_{(0,2^{m+1}-1)}$.* For $i \in [m+1]$, $j \in [n]$, the circuit $C_{(1,i,j),\rightarrow}$, given main input $\mathbf{v}$ and auxiliary input randomness $z_{1,i,j} \in \{0,1\}$, adds the vector $z_{1,i,j} \cdot 2^{i-1} \cdot \mathbf{B}_j$, where $\mathbf{B}_j$ is the j-th column of the basis $\mathbf{B}$. Note that this step uses $\ell_{\mathcal{L}} := (m+1) \cdot n$ circuits, where each uses a single bit of randomness.

2. *Stabilizing the coordinate vectors to $\mathbb{Z}^n_{(-2^m,2^m-1)}$, by subtraction.* The circuit $C_{(2,1),\rightarrow}$, which is deterministic and does not use randomness, given input $\mathbf{v}$, outputs $\mathbf{v} - 2^m \cdot \sum_{j\in[n]} \mathbf{B}_j$. This step uses a single deterministic circuit.

3. *Adding approximated Gaussian noise.* For $i_1 \in [\kappa]$, $i_2 \in [\beta+1]$, $i_3 \in [n]$, the circuit $C_{(3,i_1,i_2,i_3),\rightarrow}$, given input $\mathbf{v}$ and randomness $z_{(3,i_1,i_2,i_3)} \in \{0,1\}$, adds the vector

$$z_{(3,i_1,i_2,i_3)} \cdot 2^{i_2-1} \cdot e_{i_3} \ ,$$

where for $j \in [n]$, the vector e_j is the j-th standard basis vector, i.e., $e_j := (0_1, \cdots, 0_{j-1}, 1_j, 0_{j+1}, \cdots, 0_n)^T$. Note that this step uses $\ell_G := \kappa \cdot (\beta + 1) \cdot n$ circuits, where each uses a single bit of randomness.

4. *Stabilizing the Gaussian noise by subtraction.* The circuit $C_{(4,1),\rightarrow}$, which is deterministic and does not use randomness, given input $\mathbf{v}$, outputs

$$\mathbf{v} - \left(\frac{\kappa}{2} \cdot \left(2 \cdot 2^\beta - 1 \right) (1_1, 1_2, \cdots, 1_n)^T \right) \ .$$

This step uses a single deterministic circuit. Note that κ is even and $\kappa/2$ is an integer.

We defined what are the forward computations (that is, for every $i \in [\ell]$, the circuit $C_{i,\rightarrow}$), and the definition of the circuits $C_{i,\leftarrow}$ follows:

1. *Inverses of first $\ell_{\mathcal{L}}$ circuits.* For $i \in [m+1]$, $j \in [n]$, the circuit $C_{(1,i,j),\leftarrow}$, given main input $\mathbf{v}$ and auxiliary input randomness $z_{1,i,j} \in \{0,1\}$, subtracts the vector $z_{1,i,j} \cdot 2^{i-1} \cdot \mathbf{B}_j$ instead of adding it.

2. *Inverse of circuit $\ell_{\mathcal{L}} + 1$.* The circuit $C_{(2,1),\leftarrow}$, given input $\mathbf{v}$, adds the vector $2^m \cdot \sum_{j\in[n]} \mathbf{B}_j$ instead of subtracting it.

3. *Inverses of the circuits $\ell_{\mathcal{L}} + 1 + i$, for $i \in [\ell_G]$.* For $i_1 \in [\kappa]$, $i_2 \in [\beta + 1]$, $i_3 \in [n]$, the circuit $C_{(3,i_1,i_2,i_3),\leftarrow}$, given main input $\mathbf{v}$ and auxiliary input randomness $z_{(3,i_1,i_2,i_3)} \in \{0,1\}$, subtracts the vector $z_{(3,i_1,i_2,i_3)} \cdot 2^{i_2-1} \cdot e_{i_3}$, instead of adding it.

4. *Inverse of circuit $\ell_{\mathcal{L}} + 1 + \ell_G + 1$.* The circuit $C_{(4,1),\leftarrow}$, given input $\mathbf{v}$, adds the vector

$$\left(\frac{\kappa}{2} \cdot \left(2 \cdot 2^\beta - 1 \right) (1_1, 1_2, \cdots, 1_n)^T \right) \ ,$$

instead of subtracting it.

Regarding the output of the reduction, which is a pair of invertible circuits sequences (C^0, C^1): The first circuit sequence C^0 is exactly the circuit sequence C. As for the second circuit sequence C^1, it is the same as $C = C^0$ only that we add the vector $\mathbf{t}$ in the end, and formally, we make the following single change to $C = C^0$: The last circuit pair $C_{(4,1),\rightarrow}$, $C_{(4,1),\leftarrow}$ is changed as follows to the pair $C'_{(4,1),\rightarrow}$, $C'_{(4,1),\leftarrow}$.

- $C'_{(4,1),\rightarrow}$ is the same as $C_{(4,1),\rightarrow}$, only that it further adds $\mathbf{t}$ in the end of its computation.

- The inverse $C'_{(4,1),\leftarrow}$ acts the same as $C_{(4,1),\leftarrow}$, but further subtracts $\mathbf{t}$ in the end of its computation.

It can be easily verified that both C^0 and C^1 are legal 1-invertible circuit sequences.

Properties of the output distributions of the circuits C^0, C^1. We next observe the following properties of the output distribution of the circuit sequence C, and then for the circuits C^0, C^1.

- Note that in step 1 of the circuit C, for every $j \in [n]$, it samples a uniformly random number $M_j \in \{0, 1, \cdots, 2^{m+1} - 1\}$, and adds $M_j \cdot \mathbf{B}_j$ to the input. This means that at the end of step 1, we have the sum $\sum_{j \in [n]} M_j \cdot \mathbf{B}_j$.

- The next step 2 of the algorithm is intended to move the distribution to be uniform with expectation ≈ 0, and to this end the circuit $C_{\ell_{\mathcal{L}}+1,\rightarrow}$ subtracts $2^m \cdot \sum_{j \in [n]} \mathbf{B}_j$. At the end of this step the output distribution is accordingly $\sum_{j \in [n]} (M_j - 2^m) \cdot \mathbf{B}_j$, where for every $j \in [n]$, the variables $M_j - 2^m$ are i.i.d. samples from the uniform distribution over $\{-2^m, \cdots, -1, 0, 1, \cdots, 2^m - 1\}$. In other words, at the end of step 2 we have a random lattice vector, over a uniformly random coordinates vector in the restricted set $\mathbb{Z}^n_{(-2^m, 2^m-1)}$.

- The next step 3 of the algorithm is intended to add positive approximated Gaussian noise. Observe that at the end of this step, for each $j \in [n]$ we add κ uniformly random i.i.d. samples $\bar{\beta}_{j,i_1} \in \{0, 1, \cdots, 2^{\beta+1} - 1\}$ (for $i_1 \in [\kappa]$) to the distribution from before.

- The next step is intended to stabilize as much as possible the Gaussian noise to center around 0. To this end, step 4 subtracts $\frac{\kappa}{2} (2^\beta + 2^\beta - 1)$ ($\kappa/2$ is an integer) from each of the n coordinates of $\mathbf{v}$, this can also be thought of as subtracting $\frac{\kappa}{2} \cdot 2^\beta$ followed by subtracting $\frac{\kappa}{2} \cdot (2^\beta - 1)$.

To conclude what we saw so far,

- The output distribution of C^0 is the variable $(\mathbf{v} + \mathbf{e}) \in \mathbb{Z}^n_{(-(2^{m+b} \cdot n + \kappa \cdot 2^\beta), 2^{m+b} \cdot n + \kappa \cdot 2^\beta)}$, where $\mathbf{v} \in \mathbb{Z}^n_{(-(2^{m+b} \cdot n), 2^{m+b} \cdot n)}$ is a random lattice vector with a uniformly random coordinates vector $\mathbf{a} \in \mathbb{Z}^n_{(-2^m, 2^m-1)}$, and $\mathbf{e} \in \mathbb{Z}^n_{(-\kappa \cdot 2^\beta, \kappa \cdot 2^\beta)}$ such that for each $j \in [n]$, the coordinate $\mathbf{e}_j$ is a random variable that's the sum of $\kappa/2$ i.i.d. uniformly random samples from $\mathbb{Z}_{(-2^\beta, 2^\beta-1)}$ and $\kappa/2$ i.i.d. uniformly random samples from $\mathbb{Z}_{(-(2^\beta-1), 2^\beta)}$.

- The output distribution of C^1 is the same as that of C^0 only that the vector $\mathbf{t}$ is added in the end, that is, the output distribution of C^1 is $\mathbf{v} + \mathbf{e} + \mathbf{t}$.

Computational complexity of the reduction. For each $b' \in \{0, 1\}$, $i \in [\ell]$, $Y \in \{\rightarrow, \leftarrow\}$, it is easy to verify that the circuit $C^{b'}_{i,Y}$ executes in polynomial time in its input size, which is in turn $n \cdot (s + 1)$. The construction of the circuit $C^{b'}_{i,Y}$ given the input $(\mathbf{B}, \mathbf{t}, d)$ also takes polynomial time in the input size. Finally, the amount of circuits is ℓ and each circuits size is polynomial in $n \cdot (s + 1)$, which are both polynomial in the input size. It follows that the reduction is implemented by a classical deterministic polynomial time Turing machine.

Soundness of the reduction. The next step of our proof is to show that the reduction is correct, that is, that when $(\mathbf{B}, \mathbf{t}, d) \in \prod_{\text{YES}}$ then the output distributions of the circuits, denoted $\mathcal{D}(C^0)$, $\mathcal{D}(C^1)$ respectively, are statistically close, and when the input is in $\prod_{\text{NO}}$, then the output distributions are statistically far.

We start with soundness: In case $(\mathbf{B}, \mathbf{t}, d) \in \prod_{\text{NO}}$, then $\Delta\left(\mathcal{L}_{\mathbf{B}}, \mathbf{t}\right) > d \cdot g(n)$. Recall $\mathbf{e} \in \mathbb{Z}^n_{\left(-\kappa \cdot 2^\beta, \kappa \cdot 2^\beta\right)}$ and thus $\|\mathbf{e}\| \leq \sqrt{n} \cdot \kappa \cdot 2^\beta$. Next, due to $\beta := \lfloor \log_2(B) \rfloor$ we have

$$2^\beta \leq B := \frac{g(n) \cdot d}{2 \cdot \kappa \cdot \sqrt{n}} \ ,$$

which implies $\|\mathbf{e}\| \leq \frac{g(n) \cdot d}{2}$. It follows that for any $\mathbf{v} \in \mathcal{L}_{\mathbf{B}}$, $\Delta\left(\mathcal{L}_{\mathbf{B}}, \mathbf{v} + \mathbf{e}\right) \leq \frac{g(n) \cdot d}{2}$. Also, since $\Delta\left(\mathcal{L}_{\mathbf{B}}, \mathbf{t}\right) > d \cdot g(n)$ and $\Delta\left(\mathcal{L}_{\mathbf{B}}, \mathbf{v}\right) = 0$ then

$$\Delta\left(\mathcal{L}_{\mathbf{B}}, \mathbf{v} + \mathbf{e} + \mathbf{t}\right) = \Delta\left(\mathcal{L}_{\mathbf{B}}, \mathbf{e} + \mathbf{t}\right) \geq \Delta\left(\mathcal{L}_{\mathbf{B}}, \mathbf{t}\right) - \|\mathbf{e}\| > d \cdot g(n) - \frac{g(n) \cdot d}{2} = \frac{g(n) \cdot d}{2} \ .$$

It follows that the output distributions $\mathcal{D}\left(C^0\right)$, $\mathcal{D}\left(C^1\right)$ of the circuits don't intersect at all, that is, $\left\|\mathcal{D}\left(C^0\right) - \mathcal{D}\left(C^1\right)\right\|_{TV} = 1$, as needed for soundness.

Completeness of the reduction, first part. We now turn to the part where $(\mathbf{B}, \mathbf{t}, d) \in \prod_{\text{YES}}$, and we would like to give an upper bound on the total variation distance between the output distributions of the two circuits. We have,

$$\left\|\mathcal{D}\left(C^0\right) - \mathcal{D}\left(C^1\right)\right\|_{TV} = \left\|\mathcal{D}\left(\mathbf{v} + \mathbf{e}\right) - \mathcal{D}\left(\mathbf{v} + \mathbf{e} + \mathbf{t}\right)\right\|_{TV} \ .$$

In our next step we would like to use the fact that adding a lattice vector to a uniform distribution over that lattice, does not change the distribution. However, the lattice is infinite and we cannot sample a uniformly random vector over the entire lattice. Formally, since $(\mathbf{B}, \mathbf{t}, d) \in \prod_{\text{YES}}$ then $\Delta\left(\mathcal{L}_{\mathbf{B}}, \mathbf{t}\right) \leq d$, thus there exist $\mathbf{s} \in \mathcal{L}_{\mathbf{B}}$ and $\mathbf{e}_0 \in \mathbb{Z}^n$ such that $\mathbf{t} = \mathbf{s} + \mathbf{e}_0$ and $\|\mathbf{e}_0\| \leq d$. It follows that,

$$\left\|\mathcal{D}\left(\mathbf{v} + \mathbf{e}\right) - \mathcal{D}\left(\mathbf{v} + \mathbf{e} + \mathbf{t}\right)\right\|_{TV}$$

$$\underset{\text{(triangle inequality)}}{\leq} \left\|\mathcal{D}\left(\mathbf{v} + \mathbf{e}\right) - \mathcal{D}\left(\mathbf{v} - \mathbf{s} + \mathbf{e} + \mathbf{t}\right)\right\|_{TV} + \left\|\mathcal{D}\left(\mathbf{v} + \mathbf{e} + \mathbf{t}\right) - \mathcal{D}\left(\mathbf{v} - \mathbf{s} + \mathbf{e} + \mathbf{t}\right)\right\|_{TV}$$

$$= \left\|\mathcal{D}\left((\mathbf{v}) + \mathbf{e}\right) - \mathcal{D}\left((\mathbf{v}) + \mathbf{e} + \mathbf{e}_0\right)\right\|_{TV} + \left\|\mathcal{D}\left(\mathbf{v} + (\mathbf{e} + \mathbf{t})\right) - \mathcal{D}\left(\mathbf{v} - \mathbf{s} + (\mathbf{e} + \mathbf{t})\right)\right\|_{TV}$$

$$\underset{\text{(adding variables can only reduce TV distance)}}{\leq} \left\|\mathcal{D}\left(\mathbf{e}\right) - \mathcal{D}\left(\mathbf{e} + \mathbf{e}_0\right)\right\|_{TV} + \left\|\mathcal{D}\left(\mathbf{v}\right) - \mathcal{D}\left(\mathbf{v} - \mathbf{s}\right)\right\|_{TV} \ .$$

We use the fact that we took the bound M (which is effectively the sample space of our coordinates for the vector $\mathbf{v}$) to be exponentially (in the input dimension n) larger than the vectors in $\mathbf{B}$ and $\mathbf{t}$. Formally, note that since the norm of $\mathbf{s}$ is $\leq \|\mathbf{t}\| + \|\mathbf{e}_0\| \leq \|\mathbf{t}\| + d$, then $\|\mathbf{s}\| \cdot 2^n \leq M$. It follows that the distribution $\mathbf{v} - \mathbf{s}$ has total variation distance bounded by 2^{-n} to $\mathbf{v}$, and the above sum is bounded by

$$\left\|\mathcal{D}\left(\mathbf{e}\right) - \mathcal{D}\left(\mathbf{e} + \mathbf{e}_0\right)\right\|_{TV} + 2^{-n} \ .$$

It remains to bound the total variation distance between $\mathcal{D}\left(\mathbf{e}\right)$ and $\mathcal{D}\left(\mathbf{e} + \mathbf{e}_0\right)$ for any $\mathbf{e}_0$ such that $\|\mathbf{e}_0\| \leq d$.

Completeness of the reduction, second part (moving to discrete Gaussians). We have,

$$\left\|\mathcal{D}\left(\mathbf{e}\right) - \mathcal{D}\left(\mathbf{e} + \mathbf{e}_0\right)\right\|_{TV} \underset{\text{(triangle inequality)}}{\leq} \left\|\mathcal{D}\left(\mathbf{e}\right) - \mathcal{D}\left(D_{\tilde{B}}^n\right)\right\|_{TV}$$

$$+ \left\|\mathcal{D}\left(D_{\tilde{B}}^n\right) - \mathcal{D}\left(D_{\tilde{B}}^n + \mathbf{e}_0\right)\right\|_{TV}$$

$$+ \left\|\mathcal{D}\left(D_{\tilde{B}}^n + \mathbf{e}_0\right) - \mathcal{D}\left(\mathbf{e} + \mathbf{e}_0\right)\right\|_{TV} \ ,$$

where $\tilde{B} := \lceil \kappa \cdot \tilde{\sigma} \rceil$, $\tilde{\sigma} := \sqrt{\frac{2^{2\beta+2}+2^{\beta+2}}{12}}$, and the distribution $D_{\tilde{B}}^n$ is the n-dimensional discrete truncated Gaussian distribution, as defined in Definition 3.12. We use once again the fact that adding a numerical random variable to a pair of sums does not make it easier to distinguish the sums, and thus $\left\| \mathcal{D}\left(D_{\tilde{B}}^n + \mathbf{e}_0\right) - \mathcal{D}\left(\mathbf{e} + \mathbf{e}_0\right) \right\|_{TV} \leq \left\| \mathcal{D}\left(D_{\tilde{B}}^n\right) - \mathcal{D}\left(\mathbf{e}\right) \right\|_{TV}$, and overall we have,

$$\|\mathcal{D}\left(\mathbf{e}\right) - \mathcal{D}\left(\mathbf{e} + \mathbf{e}_0\right)\|_{TV}$$

$$\leq 2 \cdot \left\| \mathcal{D}\left(\mathbf{e}\right) - \mathcal{D}\left(D_{\tilde{B}}^n\right) \right\|_{TV} + \left\| \mathcal{D}\left(D_{\tilde{B}}^n\right) - \mathcal{D}\left(D_{\tilde{B}}^n + \mathbf{e}_0\right) \right\|_{TV} \ .$$

To bound the left element, note that each of the distributions $\mathcal{D}\left(\mathbf{e}\right)$, $\mathcal{D}\left(D_{\tilde{B}}^n\right)$ is n i.i.d. samples of some distribution, and thus,

$$\left\| \mathcal{D}\left(\mathbf{e}\right) - \mathcal{D}\left(D_{\tilde{B}}^n\right) \right\|_{TV} \leq n \cdot \left\| \mathcal{D}\left(\mathbf{e}_1\right) - \mathcal{D}\left(D\left(\tilde{B}\right)\right) \right\|_{TV} \ ,$$

where $\mathbf{e}_1$ is the distribution of the first coordinate of the vector $\mathbf{e}$ (which distributes identically for all of its coordinates $i \in [n]$). Recall that $\mathbf{e}_1$ is a random variable that's the sum of $\kappa/2$ i.i.d. uniformly random samples from $\mathbb{Z}_{\left(-2^\beta, 2^\beta-1\right)}$ and $\kappa/2$ i.i.d. uniformly random samples from $\mathbb{Z}_{\left(-\left(2^\beta-1\right), 2^\beta\right)}$. By Lemma 7.1, there exists a positive absolute constant $c_0 \in \mathbb{N}$ such that $\left\| \mathcal{D}\left(\mathbf{e}_1\right) - \mathcal{D}\left(D\left(\tilde{B}\right)\right) \right\|_{TV} \leq c_0 \cdot \frac{1}{\sqrt{\kappa}}$.

Completeness of the reduction, third part (Discrete Gaussians with close centers are close in total variation distance). So far in the proof of completeness, we saw that the total variation distance between the output distributions of the circuits C^0 and C^1 is bounded by

$$2^{-n} + \frac{c_0 \cdot 2 \cdot n}{\sqrt{\kappa}} + \left\| \mathcal{D}\left(D_{\tilde{B}}^n\right) - \mathcal{D}\left(D_{\tilde{B}}^n + \mathbf{e}_0\right) \right\|_{TV} \ ,$$

and it remains to give an upper bound for the last summand. Lemma 7.2 tells us that,

$$\left\| \mathcal{D}\left(D_{\tilde{B}}^n\right) - \mathcal{D}\left(D_{\tilde{B}}^n + \mathbf{e}_0\right) \right\|_{TV} \leq \sqrt{1 - e^{-\frac{\pi \cdot \left(\|\mathbf{e}_0\|^2 + 2 \cdot \|\mathbf{e}_0\| \cdot \sqrt{n} \cdot \tilde{B}\right)}{\tilde{B}^2}}} \ .$$

Now, recall that $\|\mathbf{e}_0\| \leq d$ and $\tilde{B} := \lceil \sqrt{\kappa} \cdot \tilde{\sigma} \rceil$ for $\tilde{\sigma} := \sqrt{\frac{2^{2\beta+2}+2^{\beta+2}}{12}}$. By further remembering that $2^\beta = \Theta(B)$, we get $\tilde{B} = \Theta\left(\frac{g(n) \cdot d}{\sqrt{n\sqrt{\kappa}}}\right)$. This means that there exists a positive absolute constant $c_1 \in \mathbb{R}_{>0}$ such that $\tilde{B} \geq \frac{c_1 \cdot g(n) \cdot d}{\sqrt{n\sqrt{\kappa}}}$. We get the following lower bound,

$$e^{-\frac{\pi \cdot \left(\|\mathbf{e}_0\|^2 + 2 \cdot \|\mathbf{e}_0\| \cdot \sqrt{n} \cdot \tilde{B}\right)}{\tilde{B}^2}}$$

$$= e^{-\frac{\pi \cdot \|\mathbf{e}_0\|^2}{\tilde{B}^2}} \cdot e^{-\frac{\pi \cdot \left(2 \cdot \|\mathbf{e}_0\| \cdot \sqrt{n} \cdot \tilde{B}\right)}{\tilde{B}^2}}$$

$$\geq e^{-\frac{\pi \cdot n \cdot \kappa}{c_1^2 \cdot g(n)^2}} \cdot e^{-\frac{\pi \cdot 2 \cdot \|\mathbf{e}_0\| \cdot \sqrt{n}}{\tilde{B}}}$$

$$\geq e^{-\frac{\pi \cdot n \cdot \kappa}{c_1^2 \cdot g(n)^2}} \cdot e^{-\frac{\pi \cdot 2 \cdot n \cdot \sqrt{\kappa}}{c_1 \cdot g(n)}} \ .$$

To summarize, in case $(\mathbf{B}, \mathbf{t}, d) \in \prod_{\text{YES}}$, we got that the total variation distance between the output distributions of the circuits C^0, C^1 is bounded by

$$2^{-n} + \frac{c_0 \cdot 2 \cdot n}{\sqrt{\kappa}} + \sqrt{1 - e^{-\frac{\pi \cdot n \cdot \kappa}{c_1^2 \cdot g(n)^2}} \cdot e^{-\frac{\pi \cdot 2 \cdot n \cdot \sqrt{\kappa}}{c_1 \cdot g(n)}}} \ .$$

Now, going back to how we defined $g(n)$ and κ, we took

- $\kappa := c_\kappa \cdot n^2$ and $g(n) := c_g \cdot n^2$, for

- $c_\kappa := c_0^2 \cdot 64$, $c_g := 256 \cdot \frac{c_0}{c_1} \cdot \pi$ (this choice of $g(n)$ and κ that depends on c_0, c_1 is legal, as c_0, c_1 are both positive absolute constants in $\mathbb{R}_{>0}$).

It follows that for these parameter choices we get,

$$2^{-n} + \frac{c_0 \cdot 2 \cdot n}{\sqrt{\kappa}} + \sqrt{1 - e^{-\frac{\pi \cdot n \cdot \kappa}{c_1^2 \cdot g(n)^2}}} \cdot e^{-\frac{\pi \cdot 2 \cdot n \cdot \sqrt{\kappa}}{c_1 \cdot g(n)}}$$

$$= 2^{-n} + \frac{c_0 \cdot 2 \cdot n}{8 \cdot c_0 \cdot n} + \sqrt{1 - e^{-\frac{\pi \cdot n^3 \cdot 64 \cdot c_0^2}{c_1^2 \cdot g(n)^2}}} \cdot e^{-\frac{\pi \cdot 2 \cdot n^2 \cdot 8 \cdot c_0}{c_1 \cdot g(n)}}$$

$$= 2^{-n} + \frac{1}{4} + \sqrt{1 - e^{-\frac{\pi \cdot n^3 \cdot 64 \cdot c_0^2}{c_1^2 \cdot g(n)^2}}} \cdot e^{-\frac{\pi \cdot 2 \cdot n^2 \cdot 8 \cdot c_0}{c_1 \cdot g(n)}}$$

$$= 2^{-n} + \frac{1}{4} + \sqrt{1 - e^{-\frac{\pi \cdot n^3 \cdot 64 \cdot c_0^2}{c_1^2 \cdot \left(256 \cdot \frac{c_0}{c_1} \cdot \pi \cdot n^2\right)^2}}} \cdot e^{-\frac{\pi \cdot 2 \cdot n^2 \cdot 8 \cdot c_0}{c_1 \cdot 256 \cdot \frac{c_0}{c_1} \cdot \pi \cdot n^2}}$$

$$= 2^{-n} + \frac{1}{4} + \sqrt{1 - e^{-\frac{1}{2^{10} \cdot \pi \cdot n}}} \cdot e^{-\frac{1}{16}}$$

$$\underset{\text{(for any } n \in \mathbb{N})}{\leq} \frac{1}{2} + 2^{-n} \ .$$

$\square$

Lemma 7.1 (Discrete uniform sum is indistinguishable from discrete Gaussian). *For $a, b \in \mathbb{N}$ let $\mathcal{U}_{(-a,b)}$ the discrete uniform distribution on the set $\{-a, \cdots, -1, 0, 1, \cdots, b\}$. For $\beta, \kappa \in \mathbb{N}$ such that β is an even number, let the following variables:*

- $S_0 := \sum_{i \in [\kappa/2]} \mathcal{U}^{(i)}_{(-2^\beta, 2^\beta - 1)}$ *the random variable which is the sum of $\kappa/2$ i.i.d. samples from* $\mathcal{U}_{(-2^\beta, 2^\beta - 1)}$.

- $S_1 := \sum_{i \in [\kappa/2]} \mathcal{U}^{(i)}_{(-(2^\beta - 1), 2^\beta)}$ *the random variable which is the sum of $\kappa/2$ i.i.d. samples from* $\mathcal{U}_{(-(2^\beta - 1), 2^\beta)}$.

- *We define the total sum of the two cases $S := S_0 + S_1$.*

Then,

$$\|(S) - (D_{\tilde{B}})\|_{TV} \leq O\left(\frac{1}{\sqrt{\kappa}}\right) \ ,$$

for $\tilde{\sigma} := \sqrt{\frac{2^{2\beta+2} + 2^{\beta+2}}{12}}$, $\tilde{B} := \lceil \sqrt{\kappa} \cdot \tilde{\sigma} \rceil$.

Proof. We use Theorem 3.3. We have the following parameters:

- The expectation of $\mathcal{U}^{(i)}_{(-2^\beta, 2^\beta - 1)}$ is $\mu_{i,0} = -\frac{1}{2}$ and the expectation of $\mathcal{U}_{(-(2^\beta - 1), 2^\beta)}$ is $\mu_{i,1} = \frac{1}{2}$.

- The variance of both $\mathcal{U}^{(i)}_{(-2^\beta, 2^\beta - 1)}, \mathcal{U}_{(-(2^\beta - 1), 2^\beta)}$ is the same: $\sigma_{i,b}^2 = \frac{\left(2^{\beta+1} + 1\right)^2 - 1}{12} = \frac{2^{2\beta+2} + 2^{\beta+2}}{12} = \Theta\left(2^{2\beta}\right)$.

- The variance of the sum S is $\sigma^2 = \sum_{i \in [\kappa]} \sigma_i^2 = \kappa \cdot \sigma_i^2 = \Theta\left(\kappa \cdot 2^{2\beta}\right)$, since we are dealing with i.i.d. random variables.

- The third moment of $\mathcal{U}^{(i)}_{(-2^\beta, 2^\beta-1)}, \mathcal{U}_{(-(2^\beta-1), 2^\beta)}$ is of the same order:

$$\gamma_{i,0} = \gamma_{i,1} := \frac{\mathbb{E}_{\mathcal{U}_{(-2^\beta, 2^\beta-1)}}\left(\left|\mathcal{U}_{(-2^\beta, 2^\beta)} - \mu_i\right|^3\right)}{\sigma^3} = \Theta\left(\frac{1}{\kappa\sqrt{\kappa}}\right) \quad .$$

- The expectation of S is $\mu := \sum_{i \in [\kappa/2], b \in \{0,1\}} \mu_{i,b} = 0$.

- The sum of third moments is $\gamma := \sum_{i \in [\kappa/2], b \in \{0,1\}} \gamma_{i,b} = \Theta\left(\frac{1}{\sqrt{\kappa}}\right)$.

- The maximal absolute value that S can take is $B := \kappa \cdot 2^\beta$.

- For every $i \in [\kappa/2]$, $b \in \{0,1\}$, the random variable $S^{(i,b)} := S - \mathcal{U}^{(i,b)}$ is the the discrete uniform sum of $\kappa - 1$ uniform distributions, instead of summing all κ samples as in S.

The theorem's upper bound on the total variation distance implies,

$$\left\|(S) - \left(\mathcal{N}^{\kappa \cdot 2^\beta}\left(0, \sqrt{\kappa} \cdot \tilde{\sigma}\right)\right)\right\|_{TV}$$

$$\leq \frac{3}{2} \cdot \sigma \cdot \sum_{i \in [\kappa/2], b \in \{0,1\}} \left(\left(\gamma_{i,b} + \frac{2 \cdot \sigma_{i,b}^2}{3 \cdot \sigma^3}\right) \cdot \left\|\left(S^{(i,b)}\right) - \left(S^{(i,b)} + 1\right)\right\|_{TV}\right)$$

$$+ \left(5 + 3\sqrt{\frac{\pi}{8}}\right) \cdot \gamma + \frac{1}{\sigma \cdot 2\sqrt{2\pi}} \quad ,$$

where $\tilde{\sigma} := \sqrt{\frac{2^{2\beta+2} + 2^{\beta+2}}{12}}$. We analyze each of the three summands separately, from last to first. As for the third (and last) summand, $\frac{1}{\sigma \cdot 2\sqrt{2\pi}} \leq O\left(\frac{1}{\sqrt{\kappa} \cdot 2^{\Omega(\beta)}}\right)$ because $\sigma \leq \Omega\left(\sqrt{\kappa} \cdot 2^{\Omega(\beta)}\right)$. The second summand is $\left(5 + 3\sqrt{\frac{\pi}{8}}\right) \cdot \gamma \leq O\left(\frac{1}{\sqrt{\kappa}}\right)$ because $\gamma \leq O\left(\frac{1}{\sqrt{\kappa}}\right)$. Overall the sum of the third and second summands together is $O\left(\frac{1}{\sqrt{\kappa}}\right)$.

The first summand is a sum by itself, and we have

$$\frac{3}{2} \cdot \sigma \cdot \sum_{i \in [\kappa/2], b \in \{0,1\}} \left(\left(\gamma_{i,b} + \frac{2 \cdot \sigma_{i,b}^2}{3 \cdot \sigma^3}\right) \cdot \left\|\left(S^{(i,b)}\right) - \left(S^{(i,b)} + 1\right)\right\|_{TV}\right)$$

$$= \sum_{i \in [\kappa/2], b \in \{0,1\}} \left(\left(\frac{3}{2} \cdot \sigma \cdot \gamma_{i,b} + \frac{\sigma_{i,b}^2}{\sigma^2}\right) \cdot \left\|\left(S^{(i,b)}\right) - \left(S^{(i,b)} + 1\right)\right\|_{TV}\right)$$

$$\underset{(*)}{=} \kappa \cdot \Theta\left(\left(\frac{3}{2} \cdot \sigma \cdot \gamma_{1,0} + \frac{\sigma_{1,0}^2}{\sigma^2}\right) \cdot \left\|\left(S^{(1,0)}\right) - \left(S^{(1,0)} + 1\right)\right\|_{TV}\right)$$

$$\leq O\left(2^\beta \cdot \frac{1}{\sqrt{\kappa}} \cdot \left\|\left(S^{(1,0)}\right) - \left(S^{(1,0)} + 1\right)\right\|_{TV}\right) + O\left(\frac{1}{\kappa} \cdot \left\|\left(S^{(1,0)}\right) - \left(S^{(1,0)} + 1\right)\right\|_{TV}\right)$$

$$\leq O\left(2^\beta \cdot \frac{1}{\sqrt{\kappa}} \cdot \left\|\left(S^{(1,0)}\right) - \left(S^{(1,0)} + 1\right)\right\|_{TV}\right) \quad ,$$

where the equality $(*)$ follows because for every $i \in [\kappa/2], b \in \{0,1\}, \gamma_{i,b} = \Theta(\gamma_{1,0}), \sigma_{i,b} = \sigma_{1,0}$, and

$$\left\|\left(S^{(i,b)}\right) - \left(S^{(i,b)} + 1\right)\right\|_{TV} = \Theta\left(\left\|\left(S^{(1,0)}\right) - \left(S^{(1,0)} + 1\right)\right\|_{TV}\right) \quad .$$

Finally, by Fact 7.1, the above is bounded by $O\left(\frac{1}{\sqrt{\kappa}}\right)$.

To move to the desired distribution $D_{\tilde{B}}$, we use the triangle inequality and Fact 3.3, to get,

$$\left\|\left(\mathcal{N}^{\kappa \cdot 2^{\beta}}\left(0, \sqrt{\kappa} \cdot \tilde{\sigma}\right)\right) - \left(D_{\tilde{B}}\right)\right\|_{TV}$$

$$\leq \left\|\left(\mathcal{N}^{\kappa \cdot 2^{\beta}}\left(0, \sqrt{\kappa} \cdot \tilde{\sigma}\right)\right) - \left(\mathcal{N}^{\kappa \cdot 2^{\beta}}\left(0, \tilde{B}\right)\right)\right\|_{TV}$$

$$+ \left\|\left(\mathcal{N}^{\kappa \cdot 2^{\beta}}\left(0, \tilde{B}\right)\right) - \left(\mathcal{N}^{\tilde{B}}\left(0, \tilde{B}\right)\right)\right\|_{TV}$$

$$+ \left\|\left(\mathcal{N}^{\tilde{B}}\left(0, \tilde{B}\right)\right) - \left(D_{\tilde{B}}\right)\right\|_{TV} \leq O\left(\frac{1}{\sqrt{\kappa}}\right) \quad .$$

$\square$

In the proof above we used the following known fact about the distance between uniform sums.

Fact 7.1 (Moving a large uniform sum has little difference). *Let S and $S^{(i,b)}$ as defined in the statement of Lemma 7.1. Then,*

$$\left\|\left(S^{(i,b)}\right) - \left(S^{(i,b)} + 1\right)\right\|_{TV} \leq O\left(2^{-\beta}\right) \quad .$$

As part of this work we prove a lemma analogous to that about spheres from [GG98]. That is, discrete Gaussian distributions with (1) sufficiently close centers of mass, and (2) sufficiently large standard deviations, have the following upper bound on their total variation distance.

Lemma 7.2 (Close discrete Gaussians have bounded total variation distance). *Let $n \in \mathbb{N}$, let $B > 0$ and let $\mathbf{e}_0 \in \mathbb{Z}^n$, then,*

$$\left\|\mathcal{D}\left(D_B^n\right) - \mathcal{D}\left(D_B^n + \mathbf{e}_0\right)\right\|_{TV} \leq \sqrt{1 - e^{-\frac{\pi \cdot \left(\|\mathbf{e}_0\|^2 + 2 \cdot \|\mathbf{e}_0\| \cdot \sqrt{n} \cdot B\right)}{B^2}}} \quad .$$

Proof. We calculate a lower bound for the fidelity between the distributions, and then draw our conclusions for the total variation distance. Let us denote by

$$P := \sum_{\mathbf{e} \in \mathbb{Z}^n_{(-B, \cdots, -1, 0, 1, \cdots, B)}} e^{-\frac{\pi \cdot \|\mathbf{e}\|^2}{B^2}} \quad ,$$

the normalization factor of the n-dimensional discrete truncated Gaussian distribution. With accordance to Definition 3.12,

- The probability density function of $\mathcal{D}\left(D_B^n\right)$ is

$$\forall \mathbf{e} \in \mathbb{Z}^n_{(-B, \cdots, -1, 0, 1, \cdots, B)} : D_B^n\left(\mathbf{e}\right) := \frac{e^{-\frac{\pi \cdot \|\mathbf{e}\|^2}{B^2}}}{P} \quad .$$

- The probability density function of $\mathcal{D}\left(D_B^n + \mathbf{e}_0\right)$ is

$$\forall \mathbf{e} \in \mathbb{Z}^n_{-B, \cdots, -1, 0, 1, \cdots, B} : \left(D_B^n + \mathbf{e}_0\right)\left(\mathbf{e}\right) := \frac{e^{-\frac{\pi \cdot \|\mathbf{e} - \mathbf{e}_0\|^2}{B^2}}}{P} = D_B^n\left(\mathbf{e} - \mathbf{e}_0\right) \quad .$$

48

The fidelity between $\mathcal{D}\left(D_B^n\right)$ and $\mathcal{D}\left(D_B^n + \mathbf{e}_0\right)$ is as follows.

$$F\left(\mathcal{D}\left(D_B^n\right), \mathcal{D}\left(D_B^n + \mathbf{e}_0\right)\right) := \sum_{\mathbf{e}\in\mathbb{Z}_{(-B,\cdots,-1,0,1,\cdots,B)}^n} \sqrt{D_B^n\left(\mathbf{e}\right)\cdot D_B^n\left(\mathbf{e}-\mathbf{e}_0\right)}$$

$$= \sum_{\mathbf{e}\in\mathbb{Z}_{(-B,\cdots,-1,0,1,\cdots,B)}^n} \sqrt{\frac{e^{-\frac{\pi\cdot\|\mathbf{e}\|^2}{B^2}}}{P}\cdot\frac{e^{-\frac{\pi\cdot\|\mathbf{e}-\mathbf{e}_0\|^2}{B^2}}}{P}}$$

$$= \sum_{\mathbf{e}\in\mathbb{Z}_{(-B,\cdots,-1,0,1,\cdots,B)}^n} \frac{1}{P}\cdot e^{-\frac{\pi\cdot\left(\|\mathbf{e}\|^2+\|\mathbf{e}-\mathbf{e}_0\|^2\right)}{2\cdot B^2}}$$

$$\underset{\text{(triangle inequality)}}{\geq} \sum_{\mathbf{e}\in\mathbb{Z}_{(-B,\cdots,-1,0,1,\cdots,B)}^n} \frac{1}{P}\cdot e^{-\frac{\pi\cdot\left(\|\mathbf{e}\|^2+(\|\mathbf{e}\|+\|\mathbf{e}_0\|)^2\right)}{2\cdot B^2}}$$

$$= \sum_{\mathbf{e}\in\mathbb{Z}_{(-B,\cdots,-1,0,1,\cdots,B)}^n} \frac{1}{P}\cdot e^{-\frac{\pi\cdot\left(\|\mathbf{e}\|^2+\|\mathbf{e}\|^2+\|\mathbf{e}_0\|^2+2\cdot\|\mathbf{e}_0\|\|\mathbf{e}\|\right)}{2\cdot B^2}}$$

$$\underset{\left(\forall\mathbf{e}\in\mathbb{Z}_{(-B,\cdots,-1,0,1,\cdots,B)}^n:\|\mathbf{e}\|\leq\sqrt{n}\cdot B\right)}{\geq} \sum_{\mathbf{e}\in\mathbb{Z}_{(-B,\cdots,-1,0,1,\cdots,B)}^n} \frac{1}{P}\cdot e^{-\frac{\pi\cdot\left(2\cdot\|\mathbf{e}\|^2+\|\mathbf{e}_0\|^2+2\cdot\|\mathbf{e}_0\|\cdot\sqrt{n}\cdot B\right)}{2\cdot B^2}}$$

$$= e^{-\frac{\pi\cdot\left(\|\mathbf{e}_0\|^2+2\cdot\|\mathbf{e}_0\|\cdot\sqrt{n}\cdot B\right)}{2\cdot B^2}} \cdot \sum_{\mathbf{e}\in\mathbb{Z}_{(-B,\cdots,-1,0,1,\cdots,B)}^n} \frac{1}{P}\cdot e^{-\frac{\pi 2\cdot\|\mathbf{e}\|^2}{2\cdot B^2}}$$

$$\underset{\text{(by definition of } P)}{=} e^{-\frac{\pi\cdot\left(\|\mathbf{e}_0\|^2+2\cdot\|\mathbf{e}_0\|\cdot\sqrt{n}\cdot B\right)}{2\cdot B^2}} .$$

Our proof ends by using the known relation between fidelity and total variation distance,

$$\left\|\mathcal{D}\left(D_B^n\right) - \mathcal{D}\left(D_B^n + \mathbf{e}_0\right)\right\|_{TV} \leq \sqrt{1 - F\left(\mathcal{D}\left(D_B^n\right), \mathcal{D}\left(D_B^n + \mathbf{e}_0\right)\right)^2} .$$

$\square$

References

[Ash96]   Abhay Ashtekar. Large quantum gravity effects: Unforeseen limitations of the classical theory. *Physical review letters*, 77(24):4864, 1996.

[ATS03]   Dorit Aharonov and Amnon Ta-Shma. Adiabatic quantum state generation and statistical zero knowledge. In *Proceedings of the thirty-fifth annual ACM symposium on Theory of computing*, pages 20–29, 2003.

[BBD+97]   Adriano Barenco, Andre Berthiaume, David Deutsch, Artur Ekert, Richard Jozsa, and Chiara Macchiavello. Stabilization of quantum computations by symmetrization. *SIAM Journal on Computing*, 26(5):1541–1557, 1997.

[BCWDW01]   Harry Buhrman, Richard Cleve, John Watrous, and Ronald De Wolf. Quantum fingerprinting. *Physical Review Letters*, 87(16):167902, 2001.

[Ber19]   Manabendra N Bera. Quantifying superpositions of quantum evolutions. *Physical Review A*, 100(4):042307, 2019.

[BMM+17]   Sougato Bose, Anupam Mazumdar, Gavin W Morley, Hendrik Ulbricht, Marko Toroš, Mauro Paternostro, Andrew A Geraci, Peter F Barker, MS Kim, and Gerard Milburn. Spin entanglement witness for quantum gravity. *Physical review letters*, 119(24):240401, 2017.

[Bru14]   Časlav Brukner. Quantum causality. *Nature Physics*, 10(4):259–263, 2014.

[BWG+18]   Alessio Belenchia, Robert M Wald, Flaminia Giacomini, Esteban Castro-Ruiz, Časlav Brukner, and Markus Aspelmeyer. Quantum superposition of massive objects and the quantization of gravity. *Physical Review D*, 98(12):126009, 2018.

[BWG+19]   Alessio Belenchia, Robert M Wald, Flaminia Giacomini, Esteban Castro-Ruiz, Časlav Brukner, and Markus Aspelmeyer. Information content of the gravitational field of a quantum superposition. *International Journal of Modern Physics D*, 28(14):1943001, 2019.

[CDPV13]   Giulio Chiribella, Giacomo Mauro D'Ariano, Paolo Perinotti, and Benoit Valiron. Quantum computations without definite causal structure. *Physical Review A*, 88(2):022318, 2013.

[CGS10]   Louis HY Chen, Larry Goldstein, and Qi-Man Shao. *Normal approximation by Stein's method*. Springer Science & Business Media, 2010.

[Cos22]   Fabio Costa. A no-go theorem for superpositions of causal orders. *Quantum*, 6:663, 2022.

[CR19]   Marios Christodoulou and Carlo Rovelli. On the possibility of laboratory evidence for quantum superposition of geometries. *Physics Letters B*, 792:64–68, 2019.

[dBDE+22]   Jan de Boer, Bianca Dittrich, Astrid Eichhorn, Steven B Giddings, Steffen Gielen, Stefano Liberati, Etera R Livine, Daniele Oriti, Kyriakos Papadodimas, Antonio D Pereira, et al. Frontiers of quantum gravity: shared challenges, converging directions. *arXiv preprint arXiv:2207.10618*, 2022.

[Dio87]   Lajos Diosi. A universal master equation for the gravitational violation of quantum mechanics. *Physics letters A*, 120(8):377–381, 1987.

[Dió89]   Lajos Diósi. Models for universal reduction of macroscopic quantum fluctuations. *Physical Review A*, 40(3):1165, 1989.

[Dio14]   Lajos Diosi. Gravitation and quantummechanical localization of macroobjects. *arXiv preprint arXiv:1412.0201*, 2014.

[DSW22]   Daine L Danielson, Gautam Satishchandran, and Robert M Wald. Gravitationally mediated entanglement: Newtonian field versus gravitons. *Physical Review D*, 105(8):086001, 2022.

[FY53]   Ronald Aylmer Fisher and Frank Yates. *Statistical tables for biological, agricultural, and medical research*. Hafner Publishing Company, 1953.

[GB20]   Flaminia Giacomini and Časlav Brukner. Einstein's equivalence principle for superpositions of gravitational fields and quantum reference frames. *arXiv preprint arXiv:2012.13754*, 2020.

[GB22]   Flaminia Giacomini and Časlav Brukner. Quantum superposition of spacetimes obeys einstein's equivalence principle. *AVS Quantum Science*, 4(1), 2022.

[GCRB19]   Flaminia Giacomini, Esteban Castro-Ruiz, and Časlav Brukner. Quantum mechanics and the covariance of physical laws in quantum reference frames. *Nature communications*, 10(1):494, 2019.

[GG98]   Oded Goldreich and Shafi Goldwasser. On the limits of non-approximability of lattice problems. In *Proceedings of the thirtieth annual ACM symposium on Theory of computing*, pages 1–9, 1998.

[Gia21]   Flaminia Giacomini. Spacetime quantum reference frames and superpositions of proper times. *Quantum*, 5:508, 2021.

[GMW91]   Oded Goldreich, Silvio Micali, and Avi Wigderson. Proofs that yield nothing but their validity or all languages in np have zero-knowledge proof systems. *Journal of the ACM (JACM)*, 38(3):690–728, 1991.

[GPV08]   Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In *Proceedings of the fortieth annual ACM symposium on Theory of computing*, pages 197–206, 2008.

[Har05]   Lucien Hardy. Probability theories with dynamic causal structure: a new framework for quantum gravity. *arXiv preprint gr-qc/0509120*, 2005.

[Har07]   Lucien Hardy. Towards quantum gravity: a framework for probabilistic theories with non-fixed causal structure. *Journal of Physics A: Mathematical and Theoretical*, 40(12):3081, 2007.

[Kit95]   A Yu Kitaev. Quantum measurements and the abelian stabilizer problem. *arXiv preprint quant-ph/9511026*, 1995.

[LS13]   Matthew S Leifer and Robert W Spekkens. Towards a formulation of quantum theory as a causally neutral theory of bayesian inference. *Physical Review A*, 88(5):052130, 2013.

[MCH09]   Wayne C Myrvold, Joy Christian, and Lucien Hardy. Quantum gravity computers: On the theory of computation with indefinite causal structure. *Quantum reality, relativistic causality, and closing the epistemic circle: essays in honour of Abner Shimony*, pages 379–401, 2009.

[MP12]   Daniele Micciancio and Chris Peikert. Trapdoors for lattices: Simpler, tighter, faster, smaller. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 700–718. Springer, 2012.

[MV17]   Chiara Marletto and Vlatko Vedral. Gravitationally induced entanglement between two massive particles is sufficient evidence of quantum effects in gravity. *Physical review letters*, 119(24):240402, 2017.

[MZ04]   Michele Mosca and Christof Zalka. Exact quantum fourier transforms and discrete logarithm algorithms. *International Journal of Quantum Information*, 2(01):91–100, 2004.

[OCB12]   Ognyan Oreshkov, Fabio Costa, and Časlav Brukner. Quantum correlations with no causal order. *Nature communications*, 3(1):1092, 2012.

[Pei10] Chris Peikert. An efficient and parallel gaussian sampler for lattices. In *Annual Cryptology Conference*, pages 80–97. Springer, 2010.

[Pen96] Roger Penrose. On gravity's role in quantum state reduction. *General relativity and gravitation*, 28:581–600, 1996.

[Reg09] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. *Journal of the ACM (JACM)*, 56(6):1–40, 2009.

[Ros19] Thomas Prest-Thomas Ricosset-Mélissa Rossi. Simple, fast and constant-time gaussian sampling over the integers for falcon. 2019.

[Sho99] Peter W Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM review*, 41(2):303–332, 1999.

[SMY23] Yuuki Sugiyama, Akira Matsumura, and Kazuhiro Yamamoto. Quantum uncertainty of gravitational field and entanglement in superposed massive particles. *Physical Review D*, 108(10):105019, 2023.

[SV03] Amit Sahai and Salil Vadhan. A complete problem for statistical zero knowledge. *Journal of the ACM (JACM)*, 50(2):196–249, 2003.

[ZCPB19] Magdalena Zych, Fabio Costa, Igor Pikovski, and Časlav Brukner. Bell's theorem for temporal order. *Nature communications*, 10(1):3772, 2019.