On the Impossibility of General Parallel Fast-Forwarding of Hamiltonian Simulation

Nai-Hui Chia ✉
Rice University, Houston, TX, USA

Kai-Min Chung ✉
Academia Sinica, Taipei, Taiwan

Yao-Ching Hsieh ✉
University of Washington, Seattle, WA, USA

Han-Hsuan Lin ✉
National Tsing Hua University, Hsinchu, Taiwan

Yao-Ting Lin ✉
University of California at Santa Barbara, CA, USA

Yu-Ching Shen ✉
Academia Sinica, Taipei, Taiwan

Abstract

Hamiltonian simulation is one of the most important problems in the field of quantum computing. There have been extended efforts on designing algorithms for faster simulation, and the evolution time T for the simulation greatly affect algorithm runtime as expected. While there are some specific types of Hamiltonians that can be fast-forwarded, i.e., simulated within time $o(T)$, for some large classes of Hamiltonians (e.g., all local/sparse Hamiltonians), existing simulation algorithms require running time at least linear in the evolution time T. On the other hand, while there exist lower bounds of $\Omega(T)$ *circuit size* for some large classes of Hamiltonian, these lower bounds do not rule out the possibilities of Hamiltonian simulation with large but "low-depth" circuits by running things in parallel. As a result, physical systems with system size scaling with T can potentially do a fast-forwarding simulation. Therefore, it is intriguing whether we can achieve fast Hamiltonian simulation with the power of parallelism.

In this work, we give a negative result for the above open problem in various settings. In the oracle model, we prove that there are time-independent sparse Hamiltonians that cannot be simulated via an oracle circuit of depth $o(T)$. In the plain model, relying on the random oracle heuristic, we show that there exist time-independent local Hamiltonians and time-dependent geometrically local Hamiltonians on n qubits that cannot be simulated via an oracle circuit of depth $o(T/n^c)$, where the Hamiltonians act on n qubits, and c is a constant. Lastly, we generalize the above results and show that any simulators that are geometrically local Hamiltonians cannot do the simulation much faster than parallel quantum algorithms.

1 Introduction

Simulating a physical system with a specified evolution time is an essential approach to study the properties of the system. In particular, given a Hamiltonian H that presents the physical system of interest and the evolution time t, the goal is to use some well-studied physical system as a simulator to implement e^{-iHt} when H is time-independent or $\exp_{\mathcal{T}}\left(-i\int_0^t H(t')dt'\right)$ for time-dependent H, where $\exp_{\mathcal{T}}$ denotes the time-ordered matrix exponential. Intuitively, a simulator simulates a Hamiltonian H step by step and thus requires time at least linear in t. On the other hand, if one can use a well-studied physical system (e.g., digital or quantum computers) to simulate the Hamiltonian of interest with time significantly less than the specified evolution time, it can significantly benefit our study of physics. Following this line of thought, a fundamental question for simulating Hamiltonians is:

> *Can a simulator simulate Hamiltonians in time strictly less than the evolution time?*

This is called *fast-forwarding of Hamiltonians.* In this work, we investigate the possibility of achieving fast-forwarding of Hamiltonian using quantum computation.

It is known that quantum algorithms can fast-forward some Hamiltonians. Atia and Aharonov [3] showed that commuting local Hamiltonians and quadratic fermionic Hamiltonians with evolution time $t = 2^{\Omega(n)}$ can be *exponentially* fast-forwarded by quantum algorithms, where the Hamiltonian applies on n qubits. This result implies the existence of quantum algorithms that simulate the two classes of Hamiltonians in poly(n) time. Gu et al. [22] showed that more Hamiltonians could be exponentially or polynomially fast-forwarded, such as the exponential fast-forwarding for block diagonalizable Hamiltonians and polynomial fast-forwarding method for frustration-free Hamiltonians at low energies.

The existence of general fast-forwarding methods for Hamiltonians using quantum computers has also been studied. In particular, people investigated whether all Hamiltonians with some "succinct descriptions", such as local and sparse Hamiltonians, can be fast-forwarded. Berry et al. [6] proved no general fast-forwarding for all sparse Hamiltonians of n qubits for evolution time $t = n\pi/2$ by a reduction from computing the parity of a binary string. In particular, computing the parity of an n-bit string requires $\Omega(n)$ quantum queries, and they showed that any algorithm that simulates the corresponding Hamiltonian in time $o(n)$ will violate the aforementioned query lower bound of parity. Atia and Aharonov [3] further showed that if all 2-sparse row-computable[1] Hamiltonians with evolution time $2^{o(n)}$ can be simulated in quantum polynomial time, then **BQP = PSPACE**. In other words, the result in [3] rules out the possibility of exponential fast-forwarding for Hamiltonians with evolution time superpolynomial in the dimension under well-known complexity assumptions. Haah et al. [23] showed that there exists a piecewise constant bounded 1D time-dependent Hamiltonian[2] $H(t)$ on n qubits, such that any quantum algorithm simulating $H(t)$ with evolution time T requires $\Omega(nT)$ gates.

All these works, however, mainly considered lower bounds on the *number of gates* required for simulation, and it does not rule out the possibility that one can complete the simulation with time strictly less than t by using *parallelism*. Briefly, if many local gates in an algorithm operate on disjoint sets of input, then these gates can be applied together, and the efficiency of the algorithm is captured by the *circuit depth* instead of the number of gates. For instance, the result in [6] was based on the fact that the query complexity of parity is $\Omega(n)$ and thus,

---

[1] Given the row index, one can efficiently compute the column indices and values of the non-zero entries of the row.

[2] The Hamiltonian is 1D local, and there is a time slicing such that $H(t)$ is time-independent within each time slice. See [23] for the formal definition.

one needs $\Omega(t)$ queries to simulate the Hamiltonian evolution; however, if one runs t queries in parallel, it is possible to solve the parity problem with one layer of queries. Therefore, a direct translation of [6] does not rule out the possibility of constant depth simulation of the Hamiltonian evolution for time t by running $O(t)$ simulations in parallel.

Parallel runtime (i.e., the circuit depth in the quantum circuit model) could be another suitable notion for capturing the efficiency of the Hamiltonian simulation. Broadly speaking, any physically controllable and implementable system can be used as a simulator, so-called quantum analogue computing [17, 18]; instead of having one local interaction at each time step, a simulator that is realized by some physical system will have the whole system evolve together. From a computational perspective, a positive result of fast-forwarding Hamiltonians using parallel algorithms can imply that the simulation can be done in time strictly less than the specified evolution time given sufficient computational resources. In particular, if there exists an algorithm that simulates all Hamiltonians in time less than the simulation time t, we might be able to further reduce the runtime by recursively applying the algorithm with sufficient quantum resources. Hence, such algorithms can be a powerful tool for studying quantum physics. In fact, parallel quantum algorithms for Hamiltonian simulation have been studied and showed some advantages. Zhang et al. [39] presented a parallel quantum algorithm, whose parallel runtime (circuit depth) has a doubly logarithmic dependency on the simulation precision. Moreover, Haah et al. [23] showed that the circuit depth of their algorithm for simulating geometrically constant-local Hamiltonians can be reduced to $O(t \cdot \text{polylog}(tn/\epsilon))$ by using ancilla qubits. In the last, choosing other physical systems similar to the target Hamiltonians as simulators is possible to gain advantages, which is the idea of quantum analogue computing.

We first explore the possibility of achieving fast-forwarding of Hamiltonians using parallel quantum algorithms, i.e., quantum algorithms that have circuit depth strictly less than the simulation time. We call this *parallel fast-forwarding*. Our first goal is to address the following question:

For all sparse or local Hamiltonians, do there exist quantum algorithms that simulate the Hamiltonians with circuit depth strictly less than the required evolution time?

Furthermore, we noticed that more general simulators (in addition to quantum circuit models) are widely considered for Hamiltonian simulation, such as quantum analogue computing. So, we are also wondering about the following question.

For all sparse or local Hamiltonians, does there exist a natural simulator that simulates the Hamiltonians with evolution time strictly less than the required evolution time?

1.1 Our Results

In the work, we give negative answers to the above questions. Roughly speaking, we show that under standard cryptographic assumptions, there exists Hamiltonians that cannot be parallelly fast-forwarded by quantum computers and any simulators that are geometrically local physical systems.

We define parallel fast-forwarding as follows:

▶ **Definition 1** (Parallel fast-forwarding). *Let* **H** *be a subset of all normalized Hamiltonian ($\|H\| = 1$) and* $\mathbf{H}_n$ *be the subset of Hamiltonian in* **H** *which acts on n qubits. We say that the set* **H** *can be $(T(\cdot), g(\cdot), \varepsilon(\cdot))$-parallel fast forwarded if there exists an efficient classical algorithm $\mathcal{A}(1^n, t)$ that outputs a circuit $C_{n,t}$, i.e., $\{C_{n,t}\}$ is a uniform quantum circuit, such that for all $n \in \mathbb{N}, t \leq T(n)$, $C_{n,t}$ satisfies the following two properties.*

■ *The circuit $C_{n,t}$ has depth at most $g(t)$.*
■ *For all $H \in \mathbf{H}_n, |\psi\rangle \in \mathbb{C}^{2^n}$, the circuit $C_{n,t}(H, |\psi\rangle)$ (or $C_{n,t}^H(|\psi\rangle)$ under the oracle setting) has output state that is $\varepsilon(n)$ close to the Hamiltonian evolution outcome $e^{-iHt}|\psi\rangle$.*

In other words, there exists uniform quantum circuit $C_{n,t}$ such that for every Hamiltonian $H \in \mathbf{H}_n$, the evolution of H to time t up to some predetermined time bound $T(\cdot)$ can be simulated by C.

Compared to [3], Definition 1 focuses on C's circuit depth instead of the number of gates and requires the depth of C to be smaller than t rather than being $\mathsf{poly}(n)$. In particular, when $t = \mathsf{superpoly}(n)$, the definition in [3] can only be satisfied by a circuit C that has gate number superpolynomially smaller than t, and C that has circuit depth slightly less than t can satisfy Definition 1. Therefore, we can also interpret the no fast-forwarding theorem in [3] as refuting the possibility of achieving Definition 1 with gate number (and also circuit depth) superpolynomially smaller than $t = \mathsf{superpoly}(n)$ for a specific family of Hamiltonians. However, given that negative result, one might ask the following question:

Can we achieve parallel fast-forwarding with $g(t)$ slightly smaller than t, such as $g(t) = \sqrt{t}$?

In this work, we address the aforementioned question and show impossibility results for parallel fast-forwarding with circuit depth $g(t)$ slightly smaller than t for local or sparse Hamiltonians. Our first result is an unconditional[3] result under the oracle model.[4]

▶ **Theorem 2** (No parallel fast-forwarding for sparse Hamiltonians relative to random permutations, simplified version of Theorem 37). *Relative to a random permutation oracle over n-bit strings, for any polynomial $T(\cdot)$, there exists a family of time-independent sparse Hamiltonians $\mathbf{H}$ such that $\mathbf{H}$ cannot be $(T(\cdot), g(\cdot), \varepsilon(\cdot))$-fast forwarded for some $g = \Omega(t)$ and $\varepsilon = \Omega(1)$.*

To obtain no fast-forwarding result in the standard model,[5] we rely on cryptographic assumptions that provide hardness against low-depth algorithms. We assume the existence of *iterative parallel-hard functions*, formally defined in Definition 43. Roughly speaking, an iterative parallel-hard function is a function of the form $f(k, x) = g^{(k)}(x) := \underbrace{g(g(\ldots g(x)))}_{k \text{ times}}$, such that g is efficiently computable (by some circuit of size s), but $g^{(k)}(x)$ is not computable for circuits with depth much less than k.

With such a cryptographic assumption, we obtained the following two no-fast-forwarding theorems under the standard model.

▶ **Theorem 3** (No parallel fast-forwarding for local Hamiltonians, simplified version of Theorem 45). *Assuming the existence of iterative parallel-hard functions with size parameter $s(n)$, then for every polynomial $T(n)$, there exists a family of time-independent local Hamiltonians $\mathbf{H}$ such that $\mathbf{H}$ cannot be $(T(\cdot), g(\cdot), \varepsilon(\cdot))$-fast forwarded for some $g = \Omega(t/s(n))$ and $\varepsilon = \Omega(1)$.*

▶ **Theorem 4** (No parallel fast-forwarding for time-dependent geometrically local Hamiltonians, simplified version of Theorem 46). *Assuming the existence of iterative parallel-hard functions with size parameter $s(n)$, then for every polynomial $T(n)$, there exists a family of time-dependent geometrically local Hamiltonian H such that $\mathbf{H}$ cannot be $(T(\cdot), g(\cdot), \varepsilon(\cdot))$-fast forwarded for some $g = \Omega(t/ns(n))$ and $\varepsilon = \Omega(1)$.*

---

[3] That is, the result holds without making any computational assumptions.
[4] By the oracle model we mean that the algorithm can only access the Hamiltonian by making (quantum) queries to the oracle that encodes the description of the Hamiltonian. See Section 8 for the definition.
[5] By the standard (plain) model we mean that the algorithm is given the classical description of the Hamiltonian as input, which is the standard setting of the Hamiltonian simulation problem. Moreover, there is no oracle that can be accessed by algorithms. We will use the terms "standard model" and "plain model" interchangeably throughout this work.

Some loss in parameters are hidden in Theorem 3 and Theorem 4. Readers are referred to the full theorems in Section 9 for precise parameters.

We note that the existence of parallel-hard functions with an iterative structure is widely used in cryptography. Our definition of iterative parallel-hard functions adapts from the iterated sequential function proposed by Boneh et al. [9]. Functions of this form play a crucial role in the recent construction of verifiable delay functions (VDF) [19, 32, 35]. In contrast to its wide usage, there have not been many proposals on candidates for such iterative hard functions. Iterative squaring [33], which is probably the most widely used candidate, is not hard against quantum circuits. There are some recent attempts toward constructing iterated quantum-hard functions from isogenies [12, 20], but these assumptions are much less well-studied.

As a concrete instantiation of our iterated parallel-hard function, we adopted a hash chain, which is also widely assumed to be hard to compute within low depth. In Section 6, we justify the quantum parallel hardness of the hash chain by showing a depth lower bound of computing the hash chain in the quantum random oracle model [10].

Our results in Theorem 2, Theorem 3, and Theorem 4 imply that no quantum algorithm can simulate certain families of local or sparse Hamiltonians with circuit depth polynomially smaller than t. For instance, suppose $t = n^c$ for some constant c and $s(n) = n^2$, then by Theorem 3, no quantum algorithm can simulate the local Hamiltonians with circuit depth smaller than t^{c-2}.

Since local Hamiltonians are sparse, Theorem 3 also implies no parallel fast-forwarding of sparse Hamiltonians in the standard model. Finally, Theorem 4 and Theorem 3 are incomparable due to the fact that the Hamiltonians in Theorem 4 are time-dependent and the depth lower bound has a factor of n.

It is worth noting that the results above show no parallel fast-forwarding when using "quantum circuits" as simulators, which does not directly imply hardness results when considering other physical systems as simulators. Especially, choosing physical systems that are similar to the Hamiltonians to be simulated is possible to gain advantages, and physical systems naturally evolve the whole system together instead of applying local operators one by one. Therefore, it is nontrivial whether similar results hold for other simulators. Fortunately, we are able to generalize Theorem 4 and Theorem 3 to show that natural simulators that are geometrically local Hamiltonians cannot do much better than quantum circuits.

▶ **Theorem 5** (No fast-forwarding for local Hamiltonians with natural simulators, simplified version of Corollary 55). *Assuming the existence of iterative parallel-hard function with size parameter $s(n)$, then for every polynomial $T(n)$, there exists a family of time-independent local Hamiltonians $\mathbf{H}$ over $\widetilde{O}(n)$ qubits satisfying the following. For any geometrically constant-local Hamiltonian H_B acting on $\mathrm{poly}(n)$ qubits, using H_B to simulate any $H_A \in \mathbf{H}$ for any evolution time $t \in [0, s(n)T(n)]$ needs an evolution time at least $(t/2s(n) - O(s(n)))/\mathrm{polylog}(tn)$.*

▶ **Theorem 6** (No fast-forwarding for geometrically local Hamiltonians with natural simulators, simplified version of Corollary 54). *Assuming the existence of iterative parallel-hard functions with size parameter $s(n)$, then for every polynomial $T(n)$, there exists a family of time-dependent geometrically local Hamiltonians $\mathbf{H}$ over $\widetilde{O}(n)$ qubits satisfying the following. For any geometrically constant-local Hamiltonian H_B acting on $\mathrm{poly}(n)$ qubits, using H_B to simulate any $H_A \in \mathbf{H}$ for any evolution time $t \in [0, ns(n)T(n)]$ needs an evolution time at least $\left(\frac{t}{ns(n)} - O(ns(n)) - \mathrm{polylog}(n)\right)/\mathrm{polylog}(tn)$.*

2 Technical Overview

The main idea

Our idea is to reduce some tasks that have a circuit or query depth lower bounds (i.e., parallel-hard problems) to simulating specific Hamiltonians with evolution time t, such that the existence of parallel fast-forwarding of the Hamiltonians will contradict the circuit depth lower bound and also violate the parallel hardness of the task. For instance, one can reduce parity, which is not in $\mathbf{QNC^0}$ (the class of all constant-depth bounded fan-in circuits), to simulate a corresponding Hamiltonian H with some time t, such that e^{-iHt} outputs the parity of the input. Along this line, if e^{-iHt} can be implemented by a constant-depth quantum circuit, we can compute parity – this violates the quantum circuit lower bound on parity! Following the same idea, one can also derive some no-go results for parallel fast-forwarding from unstructured search, where the k-parallel quantum query complexity is $\Theta(\sqrt{N/k})$, where k-parallel means each "query layer" can have k queries in parallel [24, 36].

However, there are several challenges: First, those above-mentioned parallel-hard problems can be solved in depth smaller than the input size. This could result in a Hamiltonian simulation in which the evolution time is smaller than the number of qubits. Although this might still lead to an impossibility result for parallel fast-forwarding of an $o(n)$ evolution time, parallel fast-forwarding algorithms for such a short evolution time seem not that useful. In fact, to the best of our knowledge, it is not easy to find a problem that can be computed in quantum polynomial time while having a quantum depth strictly greater than the input size using $\text{polylog}(n)$ parallel queries. So, one technical contribution of our work is finding such problems and proving their quantum depth.

Second, finding appropriate reductions from the parallel-hard problems to Hamiltonians of our interest and preserving the input size and the quantum depth is also challenging. Note that we are focusing on sparse or local Hamiltonians with evolution time, a polynomial in the number of qubits. One intuitive approach is trying the circuit-to-Hamiltonian reduction in [25, 30]. Briefly, the reduction uses a t-depth circuit on n qubits to simulate a local Hamiltonian on $n + t$ qubits with time t, where the additional t qubits are for the "clock register". This, as mentioned above, has an evolution time smaller than the number of qubits. In this work, we find reductions that map a d-depth n-qubit quantum computation with $d = \text{poly}(n)$ to a local or sparse Hamiltonian with the number of qubits and evolution time "close to" n and t respectively.

Another challenge is that we need the parallel-hard problem as an iterative structure to show no parallel fast-forwarding theorems. More specifically, our goal is to prove that some Hamiltonians cannot be parallel fast-forwarded with *any evolution time in the specified range*. Therefore, we might need a sequence of parallel-hard problems such that there are corresponding parallel-hard problems *for all t* in the range. In addition, given a parallel-hard problem with an iterative structure, it is not trivial how to reduce it to one Hamiltonian H with different evolution times t such that simulating H for different t gives the corresponding answers.

Parallel hardness of the underlining assumptions

One candidate for parallel-hard problems with an iterative structure of our purpose is the *hash chain*. Roughly speaking, let $\mathcal{X}$ be a finite set and $h : \mathcal{X} \to \mathcal{X}$ be a hash function. An s-chain of h is a sequence $x_0, x_1, \ldots, x_s \in \mathcal{X}$ such that $x_{i+1} = h(x_i)$ for any $i \in [s-1]$. Given quantum oracle access to h, the goal of the algorithm is to find an s-chain. Classically, it was

proven that classical algorithms require query depth of at least s to output an s-chain with constant probability. A similar result also holds for quantum algorithms that make quantum queries to the hash function [16]. Along this line, the hash chain seems ideal for our purpose because s can be a polynomial in $\log(|\mathcal{X}|)$ and has the iterative parallel hardness.

However, a hash function is generally irreversible, and this fails standard approaches for reducing the problem to Hamiltonian simulation. Briefly, one encodes h as a Hamiltonian H such that evaluating h is equivalent to applying e^{-iH}. Since e^{-iH} is a unitary that is reversible, evaluating h also needs to be reversible. Here, we give *permutation chain* and *twisted hash chain* that are iteratively parallel-hard and the underlining function is reversible. However, the reversibility imposes another challenge, as the ability to query the inverse of the permutation breaks the known composed oracle techniques used to prove the hardness of hash chain [16][6]. Therefore we tailored a two-step-hybrid argument to prove the hardness of the random permutation chain with the ability to query the inverse of the permutations.

Note that for oracle lower bounds of parallel query algorithms, while [24] gives optimal bounds by generalizing the adversary method, it is notoriously hard to find the suitable adversary matrices. Therefore we derive the query lower bounds for our problems by crafting a hybrid argument and using the compressed oracle technique [38] respectively.

2.1 No parallel fast-forwarding for sparse Hamiltonians relative to random permutation oracle

We first introduce the permutation chain and demonstrate how to prove Theorem 2 via the *graph-to-Hamiltonian reduction* based on the permutation chain. This shows no parallel fast-forwarding for sparse matrices relative to a random permutation oracle.

Permutation chain

One of the reversible parallel-hard problem we formulated is the *permutation chain*. In this problem, we are given as inputs q permutations of $N := 2^n$ elements $\Pi_1, \Pi_2, \ldots, \Pi_q$.[7] Let $S\Pi$ be the unitary that enables one to query to each Π_i and their inverses in superposition. Let $\bar{x}_1 = 1$ and $\bar{x}_{i+1} = \Pi_i(\bar{x}_i)$ so that $\bar{x}_{q+1} = \Pi_q(\ldots \Pi_2(\Pi_1(1)))$. With q queries to $S\Pi$, it is easy to calculate $\bar{x}_i$, while we prove that it is only possible to calculate $\bar{x}_i$ with probability $O(q\sqrt{k/N})$ using $\lfloor (q-1)/2 \rfloor$ k-parallel queries[8] to $S\Pi$. Therefore if we have $q, k = O(\text{polylog}(N))$, the success probability is negligible in n, even when k is larger than q and having access to the inverses of $\Pi_1, \Pi_2, \ldots, \Pi_q$.

To bound the success probability, we employed a two-step hybrid. First, we show that we can replace each $S\Pi$ with $S\tilde{\Pi}$. $S\tilde{\Pi}$ is a set of functions that return zeros almost everywhere except at $\{\bar{x}_i\}$, where they behave the same as $S\Pi$ (see Figure 2(a)(b)). We prove that we can approximately simulate one call to a random $S\Pi$ with two calls to $S\tilde{\Pi}$. Now, $S\tilde{\Pi}$ looks like a constant zeros function, we can erase some of its values without getting caught. In the second step, we show that we can release the $\tilde{\Pi}_i$'s on a finely controlled schedule, with only negligible change in the output probability. Define $\Pi^\perp$ to be a constant zero function. Define $S\tilde{\Pi}_\ell$ to be the unitary corresponding to $\tilde{\Pi}_1, \tilde{\Pi}_2, \ldots, \tilde{\Pi}_\ell, \Pi^\perp, \ldots$, i.e., all but the first ℓ permutations are erased (see Figure 1). We show that if we replace the first k-parallel queries

---

[6] One can use the technique in [37] to convert random permutations to random functions, but the conversion only works when the algorithm has no access to the inverse oracle.

[7] They can be viewed as a special case of one permutation of qN elements.

[8] k-parallel means each "query layer" can have k queries in parallel

of $S\tilde{\Pi}$ with $S\tilde{\Pi}_1$, second k-parallel queries of $S\tilde{\Pi}$ with $S\tilde{\Pi}_2$, third k-parallel queries of $S\tilde{\Pi}$ with $S\tilde{\Pi}_3$, etc, we can only be caught with negligible probability. Intuitively, this is because while we are at the i-th query layer, it is hard to find any non-zero values of $\tilde{\Pi}_{i+1}, \ldots, \tilde{\Pi}_q$. Therefore, if an algorithm only makes $q-1$ queries to $S\tilde{\Pi}$, we can replace the queries with $S\tilde{\Pi}_1, S\tilde{\Pi}_2, \ldots, S\tilde{\Pi}_{q-1}$. It is impossible to find $\bar{x}_{q+1}$ with non-negligible probability since these oracles do not have information of $\tilde{\Pi}_q$.

Graph-to-Hamiltonian reductions

The purpose of graph-to-Hamiltonian reduction is using quantum walk on a line [14] to solve the permutation chain. Briefly, we use a graph to encode the permutation chain and let Hamiltonian H be the adjacency matrix that represents the graph. Then, the time evolution operator e^{-iHt} helps to find the solution of permutation chain. Therefore, a low-depth Hamiltonian simulation algorithm for H could result in breaking the hardness of permutation chain. This gives our first impossibility result of parallel fast-forwarding for sparse Hamiltonians.

Let $\Pi_1, \Pi_2, \ldots \Pi_L$ be L permutations over N elements. We use a graph with $N(L+1)$ vertices in which each vertex labelled by (j, x) to record the permutation chain, where $j \in \{0, 1, \ldots, L\}$ and $x \in [N]$. The vertices (j, x) and $(j+1, x')$ are adjacent if and only if $x' = \Pi_{j+1}(x)$. The construction of the graph has followings properties. First, the graph consists of N disconnected line because each Π_j is a permutation. Second, each vertex (q, x) that connects to $(0, x_0)$ satisfies $x_q = \Pi_q(\Pi_{q-1} \cdots (\Pi_1(x_0)))$. To solve the permutation chain problem, we start from the vertex $(0, x_0)$ and walk along the connected line. When stopping at a vertex (q, x_q), the pair (x_q, x_0) would be a solution of permutation chain. It is obvious that the adjacency matrix of the corresponding graph is *sparse*. We let the Hamiltonian H determining the dynamics of the walk be the adjacency matrix of the graph, and our goal is to find (x_q, x_0) by simulating e^{-iHt} given *sparse access* to H.

There are two main challenges for building such a reduction: First, we need to implement the sparse oracle access to the corresponding Hamiltonian. This requires oracle access to the permutation and inverse permutation oracle. More specific, we need to implement two oracles that are used in the Hamiltonian simulation algorithm to execute the quantum walk. The first one is the entry oracle $\mathcal{O}_H$, which answers the element value of H when queried on the matrix index. The second one is the sparse structure oracle $\mathcal{O}_L$, which answers the indices of the nonzero entries when queried on the row index. To implement $\mathcal{O}_H$, it is equivalent to checking if two vertices (j, x) and $(j+1, x')$ are adjacency. It can be done by querying $\Pi_{j+1}(x)$. To implement $\mathcal{O}_L$, it is equivalent to finding the vertices that are adjacent to (j, x). Finding $(j+1, x')$ adjacent to (j, x) can be done by querying $\Pi_{j+1}(x)$, but finding $(j-1, x'')$ needs to query $\Pi_j^{-1}(x)$. Hence, we need to consider the security of permutation chain when the inversion oracle Π_j^{-1} is given to the adversary. We bypass this challenge by showing that the our permutation chain is secure against quantum adversaries even if inverse permutation oracle is given as we previously discussed.

Second, we need to show that the simulation algorithm is able to walk fast enough so that simulating H for evolution time close to the length of the chain gives the solution to the permutation chain. To be more precise, we aim to design the system such that after walking for time t, it reaches the vertex further than t with high probability. Recall that H determining the dynamics of the walk is the adjacency matrix of the graph corresponding to the permutation chain. We observe that for such quantum walk system, it indeed reaches some points beyond t for the walking time t with high probability. At any time t, the system is described by the quantum state $e^{-iHt}|0, x_0\rangle$. The probability of stopping on the vertex

(q, x_q) at time t is $P(q) = \left| \langle q, x_q | e^{-iHt} | 0, x_0 \rangle \right|^2$. We have $\left| \langle q, x_q | e^{-iHt} | 0, x_0 \rangle \right| = q J_q(2t)/t$ for $t \in [0, L/2]$, where $J_q(\cdot)$ is the q-th order Bessel function [14]. By the properties of Bessel function, we show that $\sum_{q=\lceil t \rceil}^{L} P(q) = O(1)$, which means that the probability of stopping at a vertex (l, x_l) such that $l > t$ is high. As a result, it breaks the hardness of permutation chain if e^{-iHt} can be implemented with $o(t)$ queries.

2.2 No parallel fast-forwarding for (geometrically) local Hamiltonians in the plain model

To show no fast-forwarding of (geometrically) local Hamiltonians in the plain model, the combination of the permutation chain and the graph-to-Hamiltonian reduction used in Section 2.1 might be insufficient. First, it is unclear how to instantiate random permutation oracle. In addition, even if we can translate the permutation chain to a parallel-hard quantum circuit in the plain model, the graph-to-Hamiltonian reduction inherently provides sparse oracle access to the Hamiltonian from oracle access to the permutation chain. However, we need to have the full classical descriptions of each local term for simulating local Hamiltonians.

Observing these difficulties, we introduce the twisted hash chain and the circuit-to-Hamiltonian reduction for proving Theorem 3 and Theorem 4.

Twisted hash chain

In order to implement a reversible operation (or a permutation), we follow the idea of the Feistel network [29]. Roughly speaking, the Feistel network is an implementation of block ciphers by using cryptographic hash functions. By means of chaining quantum query operators as in Figure 3, the outputs in each layer satisfy $x_i = H(x_{i-1}) \oplus x_{i-2}$. Therefore, we can think of it as a "quantum version" of the Feistel networks. Informally, the goal of the algorithm is to output the head and tail of a chain of length $q + 1$ by using at most q depth of queries.

For proving the parallel hardness, we use the compressed oracle technique by Zhandry [38]. In particular, the analysis is undergone in the framework of Chung et al. [16] where they generalize the technique to the parallel query model. Our proof is inspired by the parallel hardness of the standard hash chain proven in [16]. For technical reasons, the challenge is the following: in the twisted hash chain problem, the algorithm is not required to output *all* elements of the chain and their hash values. Therefore, we cannot directly apply the tools provided in [16]. In addition, we cannot simply ask the algorithm to spend extra queries for outputting the hash values since this would lead to a trivial bound (we call the extra queries for generating the whole chain the "verification" procedure). Instead, we need a more fine-grained analysis of the verification procedure. First, we notice that since $x_i = H(x_{i-1}) \oplus x_{i-2}$, the verification requires *sequential* queries. Therefore, unlike Theorem 5.9 in [16] where the verification procedure only requires parallel queries, the analysis for our purpose is more involved.

We bypass the aforementioned issue by reduction. Suppose there is an algorithm $\mathcal{A}$ outputs x_0, x_q, x_{q+1} such that $x_0, \ldots, x_{q+1}$ form a $(q + 1)$-chain by making q k-parallel queries. Then we can construct a reduction $\mathcal{B}$ which first runs $\mathcal{A}$ and obtain x_0, x_q, x_{q+1}. Next, $\mathcal{B}$ starts with x_0, x_{q+1} and queries each element of the chain iteratively *in parallel* until approaching x_{q-1}, x_{2q}. If $\mathcal{A}$ successfully outputs a $(q + 1)$-chain, then it implies that $\mathcal{B}$ also outputs the *complete* $(2q + 1)$-chain with hash values but $H(x_{2q+1})$ by making a total of $2q$ k-parallel queries. As a result, it remains to analyze the success probability of making the last additional query on x_{2q+1}. In this way, it significantly simplifies the proof.

Circuit-to-Hamiltonian reductions

For our results in the plain model, we leverage the power of the random oracle heuristic. From the parallel hardness of twisted hash chain, we can obtain a heuristically parallel-hard circuit that preserves the iterative structure. Evaluation of this circuit to large depth directly translates to computing a hash chain of large length, which is assumed to be hard for low depth circuit. To translate the hardness to a no parallel fast-forwarding result, we embed the computation of the circuit to a Hamiltonian via two different approaches.

To embed circuit computation to a time-independent Hamiltonian, we use the technique from Nagaj [30], which demonstrate how to transform a circuit computation with size T to a Hamiltonian evolution problem of time $O(T \log T)$. In our work, we make two major modification upon Nagaj's technique. First, we observed that Nagaj's technique fits well with our iterated structure of circuit. At a high level, simulating Hamiltonian obtained from Nagaj's compiler can be interpreted as a quantum walk on a line, where each point on the line correspond to a computation step/gate of the circuit. Again by the detailed analysis on Bessel function that we used in the graph-to-Hamiltonian reduction, we observe that we can obtain a "depth $O(t)$" intermediate state of computing C by evolving H for time $O(t)$. This not only gives a better fast-forward lower bound, but also allows us to obtain a Hamiltonian that is hard to fast-forward on *every* evolution time within time bound T. Second, Nagaj's construction gives a Hamiltonian of $O(n + T)$-qubits, where n is the circuit input size and T is the circuit size. This is an issue because it restricts our no fast-forwarding results to evolution times small than the Hamiltonian size. We overcome this by introducing a new design for the clock state via the Johnson graph. Our restructured clock state allows a fine-grained tradeoff between the locality parameter and the Hamiltonian size.

For our second construction, we achieve the geometrically local property with the power of time-dependent Hamiltonians. Our idea is to use the piecewise-time-independent construction from [23], in which simulating the Hamiltonian for each time segment on the initial state behaves equivalently to applying a gate on the state. We take one step further by transforming our circuit to contain gates operating on neighboring gates only. This gives us a geometrically 2-local Hamiltonian which is hard to fast-forward. Combined with the algorithm that simulates geometrically local Hamiltonians also by [23], our result tightens the gap between upper bounds and lower bounds to a small polynomial in qubit number n.

▶ **Remark 7.** Two things worth to be noted for the two approaches in Section 2.1 and Section 2.2:

- If one can instantiate random permutations by hash functions or other algorithms without using keys, one can obtain Theorem 3 and Theorem 4 by combining the permutation chain and the circuit-to-Hamiltonian reduction.

- The combination of the twisted hash chain and the circuit-to-Hamiltonian reduction can give no parallel fast-forwarding for Hamiltonians in the random oracle model. This is similar to Theorem 2; however, Theorem 2 using the permutation chain and the graph-to-Hamiltonian reduction provides a better size of the Hamiltonians. In particular, the Hamiltonian in Theorem 2 has the number of qubits independent of the evolution time, while the Hamiltonians given from the circuit-to-Hamiltonian reduction has the number of qubits that is poly-logarithmic in the evolution time.

3   Open Questions

In this work, we showed that the existence of a parallel-hard problem with an iterative structure implies no parallel fast-forwarding of sparse and (geometrically) local Hamiltonians under cryptographic assumptions. Along this line, the first question that is natural to ask is whether there exist more Hamiltonians that have succinct descriptions and cannot be parallelly fast-forwarded under other computational assumptions.

We are also wondering whether the existence of parallel-hard problems with an iterative structure is equivalent to no parallel fast-forwarding. This is equivalent to proving or disproving that no parallel fast-forwarding results in parallel-hard problems with an iterative structure. Intuitively, One can show that the existence of Hamiltonians that cannot be parallelly fast-forwarded implies some quantum circuits that have no smaller circuit depth. This follows from the fact that if one can implement a quantum circuit with a depth smaller than the quantum simulation algorithm for the Hamiltonian, one can achieve parallel fast-forwarding. However, this task asks the algorithm to output quantum states close to $e^{-iHt}|\psi\rangle$ and thus is not a "classical computational problem" as parallel-hard problems with an iterative structure.

In addition, we want to match the upper and lower bounds for parallel fast-forwarding of Hamiltonian simulation. For instance, for geometrically local Hamiltonians, the algorithms in [23] require depth $O(t \cdot \mathrm{polylog}(tn/\epsilon))$, where n is the number of qubits and ϵ is the precision parameter. There is still a $O(1/ns(n))$ gap compared to our result in Theorem 4. Likewise, our results for sparse (Theorem 2) and local Hamiltonians (Theorem 3) have not matched the upper bounds from known quantum simulation algorithms, such as [27, 28, 39].

The questions mentioned above are to investigate the optimal quantum circuit depth for Hamiltonian simulation under certain computational assumptions. Note that the Hamiltonian simulation problem has classical inputs and quantum outputs. Inspired by this, we are wondering a more general question: *is it possible to prove quantum circuit lower bounds for complexity classes that have classical inputs and quantum outputs?* For example, can we unconditionally show quantum circuit depth lower bounds for Hamiltonian simulation or some quantum states with succinct classical descriptions? Note that although showing circuit depth lower bounds for languages is challenging and has some barriers, complexity classes with quantum outputs might have specific properties and provide new insights into showing quantum circuit depth lower bounds.

4   Preliminaries and Notation

4.1   Notation

For $n \in \mathbb{N}$, we use $[n]$ to denote the set $\{1, 2, \ldots, n\}$. The trace distance between two density matrices ρ and σ is denoted by $\Delta(\rho, \sigma) := \frac{1}{2} \|\rho - \sigma\|_1 = \frac{1}{2} \mathrm{tr} \left(\sqrt{(\rho - \sigma)^\dagger (\rho - \sigma)} \right)$. Let x_1, x_2 be n-bit strings, we use $x_1 \oplus x_2$ to denote the bitwise XOR of x_1 and x_2. The Kronecker delta is denoted by δ_{jk} where $\delta_{jk} = 0$ if $j \neq k$ and $\delta_{jk} = 1$ if $j = k$.

4.2   Hamiltonian simulation

▶ **Definition 8** (Hamiltonian simulation). *A Hamiltonian simulation algorithm $\mathcal{A}$ takes as inputs the description of the Hamiltonian H, an initial state $|\psi_0\rangle$, the evolution time $t \geq 0$ and an error parameter $\epsilon \in (0, 1]$. Let $|\widetilde{\psi_t}\rangle$ be the ideal state under the Hamiltonian H for evolution time t with the initial state $|\psi_0\rangle$. In other words, $|\widetilde{\psi_t}\rangle := e^{-iHt}|\psi_0\rangle$ for a*

time-independent H, and $|\widetilde{\psi_t}\rangle := \exp_{\mathcal{T}}\left(-i\int_0^t H(t')dt'\right)|\psi_0\rangle$ for a time-dependent H, where $\exp_{\mathcal{T}}$ *is the time-ordered matrix operator. The goal of $\mathcal{A}$ is to generate an approximation* $|\psi_t\rangle$ *of the evolved quantum state $|\widetilde{\psi_t}\rangle$ such that*

$$\Delta\left(|\psi_t\rangle\langle\psi_t|, |\widetilde{\psi_t}\rangle\langle\widetilde{\psi_t}|\right) \le \epsilon.$$

4.3 Basic quantum computation

Below, we provide a brief introduction to quantum computation. For more basics, we refer the readers to [31]. Throughout this work, we use the standard bra-ket notation.

▶ **Definition 9** (Quantum circuit model). *A quantum circuit consists of qubits, a sequence of quantum gates, and measurements. A qubit is a two-dimensional complex Hilbert space. Each qubit is associated with a* register. *A quantum gate is a unitary operator acting on quantum registers. We say a quantum gate is a k-qubit gate if it acts non-trivially on k qubits.*

▶ **Theorem 10** (Universal gate sets [11]). *There exists a* universal gate set *that consists of a finite number of quantum gates such that any unitary operator can be approximated by composing elements in the universal gate set within an arbitrary error. Furthermore, every element in the universal gate set is a one- or two-qubit gate.*

▶ **Definition 11** (Quantum circuit depth). *Given a finite-sized gate set $\mathcal{G}$, a d-depth quantum* circuit *or a quantum circuit of depth d with respect to $\mathcal{G}$ consists of a sequence of d layers of gates such that (i) each gate belongs in $\mathcal{G}$ and (ii) each gate within the same layer acts on disjoint qubits. We omit the gate set $\mathcal{G}$ when it is clear from the context.*

▶ **Definition 12** (Quantum query operator). *Given an oracle $f : \{0,1\}^n \to \{0,1\}^m$, the* query operator $\mathcal{O}_f$ *is defined as*

$$\mathcal{O}_f|x,y\rangle := |x, y \oplus f(x)\rangle.$$

▶ **Definition 13** (Parallel quantum query operator). *Given an oracle $f : \{0,1\}^n \to \{0,1\}^m$. The k-parallel query operator $\mathcal{O}_f^{\otimes k}$ is defined as*

$$\mathcal{O}_f^{\otimes k}|\mathbf{x}, \mathbf{y}\rangle := |\mathbf{x}, \mathbf{y} \oplus f(\mathbf{x})\rangle,$$

where $\mathbf{x} = (x_1, \ldots, x_k)$, $\mathbf{y} = (y_1, \ldots, y_k)$ and $f(\mathbf{x}) := (f(x_1), \ldots f(x_k))$.

4.4 Useful tools

In this subsection, we introduce several definitions and lemmas for analyzing quantum random walk in Section 7 and the clock state construction in Section 9.

4.4.1 Bessel functions

The Bessel functions of the first kind of order n are denoted by $J_n(x)$. We present the required properties of Bessel functions for our use.

▬ The integration form of the Bessel function:

$$J_n(x) = \frac{1}{2\pi}\int_{-\pi}^{\pi} dp\, e^{inp - ix\sin p} = \frac{i^n}{2\pi}\int_{-\pi}^{\pi} dp\, e^{inp - ix\cos p}. \tag{1}$$

- The relation between J_n and J_{-n}:

$$J_{-n}(x) = (-1)^n J_n(x). \tag{2}$$

- The recursion formula for integer orders:

$$J_{n+1}(x) = \frac{2n}{x} J_n(x) - J_{n-1}(x). \tag{3}$$

- The asymptotic form for large order

$$J_n(n \operatorname{sech}\xi) \sim \frac{e^{-n(\xi - \tanh \xi)}}{\sqrt{2\pi n \tanh \xi}} \tag{4}$$

suggests that when $x < |n|$, the value of $J_n(x)$ is exponentially small in n.

The following lemmas provide upper bounds for Bessel functions for large argument x.

▶ **Lemma 14** (Theorem 2 in [26]). *Let $n > -1/2$ and $\mu := (2n + 1)(2n + 3)$. For any $x > \sqrt{\mu + \mu^{2/3}}/2$, it holds that*

$$J_n^2(x) \leq \frac{4 \left(4x^2 - (2n+1)(2n+5)\right)}{\pi \left((4x^2 - \mu)^{3/2} - \mu\right)}.$$

By Lemma 14, we have the following lemma which is more convenient for our use.

▶ **Lemma 15.** *Let n be a positive integer. For any real $x \geq 2n$, it holds that*

$$J_n^2(x) \leq \frac{2}{n\pi}.\text{[9]}$$

Proof. We discuss the behavior of Bessel functions in three cases: $n = 1$, $n = 2$, and $n \geq 3$. For $n = 1$, the maximum of $J_1^2(x)$ is $0.339\ldots$ which is less than $2/\pi \approx 0.637$. For $n = 2$, the maximum of $J_2^2(x)$ is $0.237\ldots$ which is less than $1/\pi \approx 0.318$.

Now let us analyze the case in which $n \geq 3$. First, we notice that when $x > 2n$, the conditions in Theorem 14 hold. This is because $\mu + \mu^{2/3} < 2\mu$ and then

$$\frac{\sqrt{\mu + \mu^{2/3}}}{2} < \frac{\sqrt{2\mu}}{2} = \sqrt{2n^2 + 4n + \frac{3}{2}} < \sqrt{4n^2} = 2n < x,$$

where the second inequality holds when $n \geq 3$.

Now, we will finish the proof by bounding the numerator and the denominator of the RHS in Lemma 14. For the numerator, we have

$$4 \left(4x^2 - (2n+1)(2n+5)\right) < 4 \left(4x^2 - (2n+1)(2n+3)\right) = 4 \left(4x^2 - \mu\right).$$

For the denominator, we will show that

$$(4x^2 - \mu)^{\frac{3}{2}} - \mu > \frac{2}{3}(4x^2 - \mu)^{\frac{3}{2}}$$

or equivalently

$$\frac{1}{3}(4x^2 - \mu)^{\frac{3}{2}} > \mu.$$

First, since $x \geq \sqrt{2\mu}/2$, we have $4x^2 - \mu \geq \mu$. Furthermore, when $n \geq 3$ we have $\mu \geq 35$, which would imply $\frac{1}{3}\mu^{3/2} > \mu$. Hence, we conclude that $\frac{1}{3}(4x^2 - \mu)^{\frac{3}{2}} \geq \frac{1}{3}\mu^{3/2} > \mu$. Putting things together, we obtain

$$J_n^2(x) < \frac{4}{\pi} \cdot \frac{(4x^2 - \mu)}{\frac{2}{3}(4x^2 - \mu)^{3/2}} = \frac{4}{\pi} \cdot \frac{1}{\frac{2}{3}\sqrt{4x^2 - \mu}}.$$

When $x > 2n$ and $n > 3$, it holds that $4x^2 - \mu \geq 16n^2 - (4n^2 + 8n + 3) \geq 9n^2$. Therefore, we finally obtain

$$J_n^2(x) \leq \frac{4}{\pi} \cdot \frac{1}{\frac{2}{3}\sqrt{4x^2 - \mu}} \leq \frac{4}{\pi} \cdot \frac{1}{\frac{2}{3} \cdot 3n} = \frac{2}{n\pi}.$$

This finishes the proof. ◀

4.4.2 Johnson graph

▶ **Definition 16** (Johnson Graph). *For all integers $n \geq k \geq 1$, the (n, k)-Johnson graph $J_{n,k} = (V, E)$ is an undirected acyclic graph defined as follows.*

- $V \coloneqq \{S \subseteq [n] \colon |S| = k\}$, *i.e., the vertices are the k-element subsets of an n-element set.*
- $E \coloneqq \{(S_0, S_1) \colon |S_0 \cap S_1| = k - 1\}$, *i.e., there is an edge if and only if the intersection of the two vertices (subsets) contains $k - 1$ elements.*[10]

The number of vertices in $J_{n,k}$ is $\binom{n}{k}$. It was proven that for all integers $n \geq k \geq 1$, there exists a Hamiltonian path[11] in $J_{n,k}$ [2].

5 Lower Bounding Permutation Chain

▶ **Definition 17** (Permutation notations). *Here we define several notations for the later proofs. Let $\Pi_1, \Pi_2, \ldots, \Pi_q$ be permutations of N elements. Let $\Pi_1^{-1}, \Pi_2^{-1}, \ldots, \Pi_q^{-1}$ be the corresponding inverse permutations. Define the sets $[-q] \coloneqq \{-q, -q + 1, \ldots, -1\}$ and $[\pm q] \coloneqq [q] \cup [-q]$. We define the unitary $S\Pi$ as the controlled version of the above permutations as*

$$S\Pi|j, x, r\rangle \coloneqq \begin{cases} |j, x, r \oplus \Pi_j(x)\rangle & , j > 0 \\ |j, x, r \oplus \Pi_{|j|}^{-1}(x)\rangle & , j < 0 \end{cases} \tag{5}$$

where $j \in [\pm q]$ and $x, r \in [N]$.
We denote the elements of the chain by $\bar{x}_1 \coloneqq 1$ and $\bar{x}_{i+1} \coloneqq \Pi_i(\bar{x}_i)$ for all $i \in [k]$. Next, we define $\tilde{\Pi}_i$ to be the "erased" Π_i for all $i \in [q]$. Formally, $\tilde{\Pi}_i$ is defined to be the function $[N] \to [N] \cup \{0\}$ such that

$$\tilde{\Pi}_i(x) = \begin{cases} \bar{x}_i & , x = \bar{x}_{i-1} \\ 0 & , \text{otherwise.} \end{cases} \tag{6}$$

Similarly, we define the corresponding controlled unitary $S\tilde{\Pi}$. For all $i \in [k]$, define $\Delta\bar{x}_i \coloneqq \bar{x}_{i+1} - \bar{x}_i \mod N$. Note that $S\tilde{\Pi}$ can be parameterized by either $\{\bar{x}_2, \ldots, \bar{x}_{q+1}\}$ or $\{\Delta\bar{x}_1, \ldots, \Delta\bar{x}_q\}$. Denote the transformation from $S\Pi$ to $S\tilde{\Pi}$ by $S\tilde{\Pi} = F(S\Pi)$. For all $\ell \in [q]$, define the hybrid oracle $S\tilde{\Pi}_\ell$ as

---

[10] Equivalently, we can define $E \coloneqq \{(S_0, S_1) \colon |S_0 \cup S_1| = k + 1\}$.
[11] A Hamiltonian path is a path that visits every vertex in the graph exactly once. Do not confuse it with the physical quantity we want to simulate.
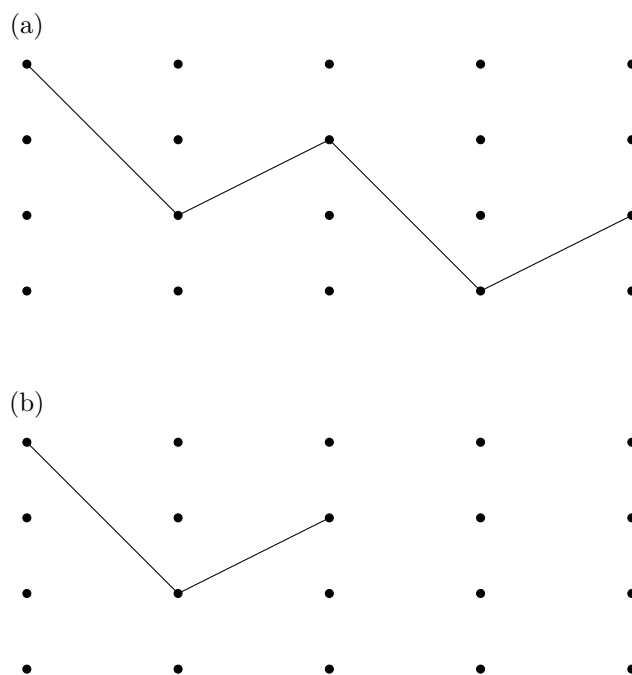
Figure 1 Schematic diagram of $S\tilde{\Pi}$ and $S\tilde{\Pi}_\ell$. (a) The permutation chain $\bar{x}_1, \bar{x}_2, \ldots \bar{x}_{q+1}$ specified by $S\tilde{\Pi}$. (b) $S\tilde{\Pi}_\ell$, where the permutations are removed after ℓ queries.

$$S\tilde{\Pi}_\ell |j, x, r\rangle := \begin{cases} |j, x, r \oplus \tilde{\Pi}_j(x)\rangle & , |j| \leq \ell, j > 0 \\ |j, x, r\rangle & , j > \ell \\ |j, x, r \oplus \tilde{\Pi}_{|j|}^{-1}(x)\rangle & , |j| \leq \ell, j < 0 \\ |j, x, r\rangle & , j < -\ell. \end{cases} \tag{7}$$

▶ **Theorem 18.** *Use the notations of Definition 17. Let q, k be integers such that $k = O(\text{polylog}(N))$ and $q = O(\text{polylog}(N))$. For any quantum algorithm $\mathcal{A}$ using $\lfloor (q-1)/2 \rfloor$ k-parallel queries to $S\Pi$, we have*

$$\mathbb{E}_{S\Pi} \left[\Pr[\mathcal{A}^{S\Pi} \text{ outputs } \bar{x}_{q+1}] \right] = O\left(q \sqrt{\frac{k}{N}} \right).$$

Before proving Theorem 18, we first introduce several lemmas as follows.

▶ **Lemma 19.** *For any $|\phi\rangle$, $|\psi\rangle$ such that $\||\phi\rangle\| = \||\psi\rangle\| = 1$ and $\||\phi\rangle - |\psi\rangle\| \leq \varepsilon$, it holds that*

$$\Delta(|\phi\rangle\langle\phi|, |\psi\rangle\langle\psi|) \leq \varepsilon.$$

Proof. The trace distance between two pure states is given by $\sqrt{1 - |\langle\phi|\psi\rangle|^2}$. The Euclidean norm of $|\phi\rangle - |\psi\rangle$ is given by $\||\phi\rangle - |\psi\rangle\| = \sqrt{(\langle\phi| - \langle\psi|)(|\phi\rangle - |\psi\rangle)} = \sqrt{2 - 2\text{Re}[\langle\phi|\psi\rangle]}$, where $\text{Re}[\cdot]$ denote the real part of a complex number.

First, it is true that $0 \leq (\text{Re}[\langle\phi|\psi\rangle] - 1)^2 + \text{Im}[\langle\phi|\psi\rangle]^2$, where $\text{Im}[\cdot]$ denote the imaginary part of a complex number. Rearranging the terms, we obtain

$$1 - |\langle\phi|\psi\rangle|^2 = 1 - (\text{Re}[\langle\phi|\psi\rangle]^2 + \text{Im}[\langle\phi|\psi\rangle]^2) \leq 2 - 2\text{Re}[\langle\phi|\psi\rangle]. \qquad \blacktriangleleft$$

▶ **Lemma 20** (*q-bin k-parallel Grover search lower bound*)**.** *Let $\mathcal{F}$ be the set of all functions f from $[qN]$ to $\{0, 1\}$ with the following promise. For all $i \in \{0, 1, \ldots, q-1\}$, it holds that $|\{x \in \{iN + 1, iN + 2, \ldots, iN + N\}: f(x) = 1\}| = 1$. Let g be the constant zero function with domain $[qN]$. Then for every algorithm that makes ℓ k-parallel queries to f (or g), the final state of the algorithm, denoted by $|\psi^f\rangle$ (or $|\psi^g\rangle$), satisfies*

$$\mathop{\mathbb{E}}_{f \leftarrow \mathcal{F}} \left[\Delta\left(|\psi^f\rangle\langle\psi^f|, |\psi^g\rangle\langle\psi^g|\right)\right] = O\left(\ell\sqrt{\frac{k}{N}}\right).$$

Proof. Let $|\psi_i^g\rangle := U_i O_g^{\otimes k} \ldots U_1 O_g^{\otimes k} U_0 |0\rangle$. For any $f \in \mathcal{F}$, let Π_f be the projector acting on the query register of the algorithm that is defined as $\Pi_f := \sum_{x:f(x)=1} |x\rangle\langle x|$. Let $\overline{\Pi}_f := I - \Pi_f$.

Notice that $(O_f^{\otimes k} - O_g^{\otimes k})\overline{\Pi}_f^{\otimes k} = 0$ because O_f and O_g are identical beyond the set of the 1-preimages. Therefore, we have

$$(O_f^{\otimes k} - O_g^{\otimes k})(I - \overline{\Pi}_f^{\otimes k}) = O_f^{\otimes k} - O_g^{\otimes k}. \tag{8}$$

Also, it holds that

$$I - \overline{\Pi}_f^{\otimes k} \leq \sum_{i=j}^{k} \Pi_{f,j}, \tag{9}$$

where $\Pi_{f,j}$ denote the operator the acts as Π_f on the register of the j-th query branch and as identity on other registers; for matrices A, B, by $A \geq B$ we mean that $A - B$ is a positive semi-definite matrix. Then by standard hybrid arguments [5], we have

$$\mathop{\mathbb{E}}_{f \leftarrow \mathcal{F}} \left[\||\psi^f\rangle - |\psi^g\rangle\|\right] \leq \sum_{i=0}^{\ell-1} \mathop{\mathbb{E}}_{f \leftarrow \mathcal{F}} \left[\|(O_f^{\otimes k} - O_g^{\otimes k})|\psi_i^g\rangle\|\right]$$

$$= \sum_{i=0}^{\ell-1} \mathop{\mathbb{E}}_{f \leftarrow \mathcal{F}} \left[\|(O_f^{\otimes k} - O_g^{\otimes k})(I - \overline{\Pi}_f^{\otimes k})|\psi_i^g\rangle\|\right]$$

$$\leq \sum_{i=0}^{\ell-1} \mathop{\mathbb{E}}_{f \leftarrow \mathcal{F}} \left[\|(O_f^{\otimes k} - O_g^{\otimes k})\| \cdot \|(I - \overline{\Pi}_f^{\otimes k})|\psi_i^g\rangle\|\right],$$

where the first equality is due to (8) and the last inequality is due to the fact that $\|A|\phi\rangle\| \leq \|A\| \cdot \||\phi\rangle\|$, where $\|A\|$ denotes the operator norm of A.

Since $\|O_f\| = \|O_g\| = 1$, by the triangle inequality we can bound it as

$$\leq 2 \sum_{i=0}^{\ell-1} \mathop{\mathbb{E}}_{f \leftarrow \mathcal{F}} \left[\|(I - \overline{\Pi}_f^{\otimes k})|\psi_i^g\rangle\|\right]$$

$$= 2 \sum_{i=1}^{\ell-1} \mathop{\mathbb{E}}_{f \leftarrow \mathcal{F}} \left[\sqrt{\langle\psi_i^g|(I - \overline{\Pi}_f^{\otimes k})|\psi_i^g\rangle}\right]$$

$$\leq 2 \sum_{i=0}^{\ell-1} \sqrt{\mathop{\mathbb{E}}_{f \leftarrow \mathcal{F}} \left[\langle\psi_i^g|(I - \overline{\Pi}_f^{\otimes k})|\psi_i^g\rangle\right]} \qquad \text{(Jensen's inequality)}$$

$$\leq 2 \sum_{i=0}^{\ell-1} \sqrt{\mathop{\mathbb{E}}_{f \leftarrow \mathcal{F}} \left[\langle\psi_i^g|\sum_{j=1}^{k} \Pi_{f,j}|\psi_i^g\rangle\right]}$$

$$\leq 2 \sum_{i=0}^{\ell-1} \sqrt{\sum_{j=1}^{k} \mathbb{E}_{f \leftarrow \mathcal{F}} [\langle \psi_i^g | \Pi_{f,j} | \psi_i^g \rangle]} \qquad \text{(linearity of expectation)}$$

$$= 2 \sum_{i=0}^{\ell-1} \sqrt{\sum_{j=1}^{k} \langle \psi_i^g | \frac{I}{N} | \psi_i^g \rangle} = 2\ell \sqrt{\frac{k}{N}},$$

where the first equality holds since $I - \overline{\Pi}_f^{\otimes k}$ is a projection operator; the third inequality holds due to (9) and the second equality holds because the probability of any $x \in [qN]$ being a 1-preimage of f is $1/N$. So we have $\mathbb{E}_{f \leftarrow \mathcal{F}}[\Pi_{f,j}] = I/N$ for every j. Finally, by Lemma 19, we have

$$\mathbb{E}_{f \leftarrow \mathcal{F}} \left[\Delta \left(|\psi^f\rangle\langle\psi^f|, |\psi^g\rangle\langle\psi^g| \right) \right] \leq 2\ell \sqrt{\frac{k}{N}}.$$

as desired.   ◀

(a) Π_i (b) $\tilde{\Pi}_i$ (c) Π_i^R (d) $\tilde{\Pi}_i'$ (e) $H(\tilde{\Pi}_i')$
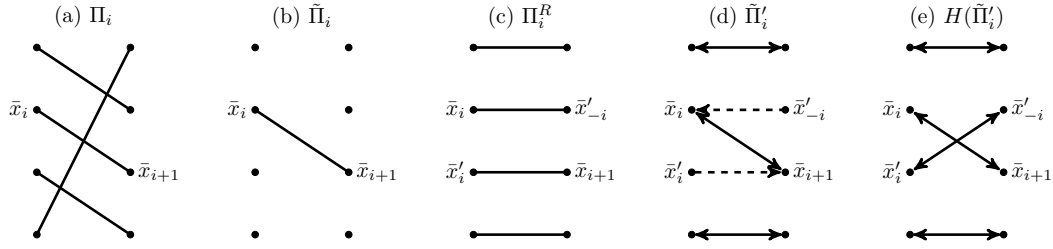
Figure 2 Schematic diagrams of permutation in the hybrid proof. (a) Π_i is the permutation given by the problem. (b) $\tilde{\Pi}_i$ maps $\bar{x}_i$ to $\bar{x}_{i+1} := \Pi_i(\bar{x}_i)$ and maps other inputs to a dummy image 0. (c) Π_i^R is a random permutation that is independent of Π_i. (d) $\tilde{\Pi}_i'$ merges $\tilde{\Pi}_i$ and Π_i^R. It maps $\bar{x}_i$ to $\bar{x}_{i+1}$, and maps other input x to $\Pi_i^R(x)$. There is a collision on inputs $\bar{x}_i$ and $\bar{x}_i' := \Pi_i^{R^{-1}}(\bar{x}_{i+1})$. When executing the "inverse" of $\tilde{\Pi}_i'$, it follows the rules of $\Pi_i^{R^{-1}}$. Note that the "inverse" is not exactly equal to $\tilde{\Pi}_i'^{-1}$. (e) The truth table of $H(\tilde{\Pi}_i')$ is equal to $\tilde{\Pi}_i'$ except that $H(\tilde{\Pi}_i')(\bar{x}_i') = \bar{x}_{-i}'$.

▶ **Lemma 21.** *Use the notations of Definition 17. Let ℓ, k be integers such that $\ell = O(\mathrm{polylog}(N))$ and $k = O(\mathrm{polylog}(N))$. For any quantum algorithm $\mathcal{A}$ using ℓ k-parallel queries to $S\Pi$, there is a quantum algorithm $\tilde{\mathcal{A}}$ using 2ℓ k-parallel queries to $S\tilde{\Pi}$ such that for all $S\tilde{\Pi}$,*

$$\mathbb{E}_{S\Pi \in F^{-1}(S\tilde{\Pi})} \left[\Pr[\mathcal{A}^{S\Pi} \neq \tilde{\mathcal{A}}^{S\tilde{\Pi}}] \right] = O\left(\ell \sqrt{\frac{k}{N}} \right).$$

Proof. Let $\Pi_1^R, \Pi_2^R, \ldots, \Pi_q^R$ be permutations of N elements. Let $S\Pi^R$ be the corresponding unitary.

We construct $\tilde{\mathcal{A}}$ as follows: it samples a uniformly random $S\Pi^R$ and runs $\mathcal{A}$ but replaces every query to $S\Pi$ with $S\tilde{\Pi}'$ constructed from uniformly random $S\Pi^R$, where $S\tilde{\Pi}'$ is defined as

$$S\tilde{\Pi}'|i, x, r\rangle := \begin{cases} |i, x, r \oplus \tilde{\Pi}_i(x)\rangle & , \tilde{\Pi}_i(x) \neq 0, i > 0 \\ |i, x, r \oplus \Pi_i^R(x)\rangle & , \tilde{\Pi}_i(x) = 0, i > 0 \\ |i, x, r \oplus \tilde{\Pi}_{|i|}^{-1}(x)\rangle & , \tilde{\Pi}_{|i|}^{-1}(x) \neq 0, i < 0 \\ |i, x, r \oplus \Pi_{|i|}^{R^{-1}}(x)\rangle & , \tilde{\Pi}_{|i|}^{-1}(x) = 0, i < 0. \end{cases} \qquad (10)$$

One query to $S\tilde{\Pi}'$ can be constructed from two queries to $S\tilde{\Pi}$ coherently by doing the obvious classical calculation and uncomputing the garbage. Therefore $\tilde{\mathcal{A}}$ uses 2ℓ queries to $S\tilde{\Pi}$ as required.

Now we prove that it is very hard to distinguish $\mathcal{A}$ from $\tilde{\mathcal{A}}$. This is done by a reduction to the hardness of the modified Grover's search algorithm in Lemma 20.

For all $i \in [q]$, define $\bar{x}'_i$ to be x such that $\tilde{\Pi}'(x) = \bar{x}_{i+1}$ and $x \neq \bar{x}$. For all $i \in [-q]$, define $\bar{x}'_i$ to be x such that $\tilde{\Pi}'^{-1}_{|i|}(x) = \bar{x}_{|i|}$ and $x \neq \bar{x}_{|i|+1}$. Note that $\bar{x}'_i$ does not exist if $\Pi^R_i(\bar{x}_i) = \tilde{\Pi}_i(\bar{x}_i)$.

Note that each $\tilde{\Pi}'_i$ looks like a random permutation except on the collisions $\{\bar{x}'_i\}$. We define a function H which relates these similar $S\Pi$ and $S\tilde{\Pi}'$:

$$H(S\tilde{\Pi}')|i,x,r\rangle := \begin{cases} |i,x,r \oplus \tilde{\Pi}_i(x)\rangle & , \tilde{\Pi}_i(x) \neq 0, i > 0 \\ |i,x,r \oplus \Pi^R_i(x)\rangle & , \tilde{\Pi}_i(x) = 0, x \neq \bar{x}'_i, i > 0 \\ |i,x,r \oplus \bar{x}'_{-i}\rangle & , x = \bar{x}'_i, i > 0 \\ |i,x,r \oplus \tilde{\Pi}^{-1}_{|i|}(x)\rangle & , \tilde{\Pi}^{-1}_{|i|}(x) \neq 0, i < 0 \\ |i,x,r \oplus \Pi^{R^{-1}}_{|i|}(x)\rangle & , \tilde{\Pi}^{-1}_{|i|}(x) = 0, x \neq \bar{x}'_i, i < 0 \\ |i,x,r \oplus \bar{x}'_{-i}\rangle & , x = \bar{x}'_i, i < 0. \end{cases} \tag{11}$$

It is easy to check that for all $S\tilde{\Pi}'$, $H(S\tilde{\Pi}')$ is a valid $S\Pi$, and for a given $S\Pi$, there are N^q elements in $H^{-1}(S\Pi)$. It is also easy to verify that for a fixed $S\tilde{\Pi}$, different $H^{-1}(S\Pi)$ partitions all possible $S\tilde{\Pi}'$.

Let $\rho_{S\Pi}$ be the final density matrix of $\mathcal{A}^{S\Pi}$. Let $\rho_{S\tilde{\Pi}'}$ be the final density matrix of $\tilde{\mathcal{A}}^{S\tilde{\Pi}}$ with a fixed $S\Pi'$. By the strong convexity of trace distance, we have

$$\Delta\left(\mathop{\mathbb{E}}_{S\Pi \in F^{-1}(S\tilde{\Pi})}[\rho_{S\Pi}], \mathop{\mathbb{E}}_{S\tilde{\Pi}' \in H^{-1}(F^{-1}(S\tilde{\Pi}))}[\rho_{S\tilde{\Pi}'}]\right) \leq \mathop{\mathbb{E}}_{S\Pi \in F^{-1}(S\tilde{\Pi})}\left[\Delta\left(\rho_{S\Pi}, \mathop{\mathbb{E}}_{S\tilde{\Pi}' \in H^{-1}(S\Pi)}[\rho_{S\tilde{\Pi}'}]\right)\right]. \tag{12}$$

Finally, we prove that $\Delta(\rho_{S\Pi}, \mathbb{E}_{S\tilde{\Pi}' \in H^{-1}(S\Pi)}[\rho_{S\tilde{\Pi}'}]) = O\left(\ell\sqrt{\frac{k}{N}}\right)$ for all $S\Pi$ by a reduction to the modified Grover search problem. Consider a Grover oracle G defined in Lemma 20 that might be f or g. Given free calls to $S\Pi$, we use two calls to G to construct an oracle $S\Pi^G$ which equals $S\Pi$ when $G = g$ and equals a random $S\tilde{\Pi}'$ when $G = f$. The construction is as follows:

$$S\Pi^G|i,x,r\rangle := \begin{cases} |i,x,r \oplus \tilde{\Pi}_i(x)\rangle & , \tilde{\Pi}_i(x) \neq 0, i > 0 \\ |i,x,r \oplus \Pi_i(x)\rangle & , \tilde{\Pi}_i(x) = 0, G((i-1)N+x) = 0, i > 0 \\ |i,x,r \oplus \bar{x}_{i+1}\rangle & , \tilde{\Pi}_i(x) = 0, G((i-1)N+x) = 1, i > 0 \\ |i,x,r \oplus \tilde{\Pi}^{-1}_{|i|}(x)\rangle & , \tilde{\Pi}^{-1}_{|i|}(x) \neq 0, i < 0 \\ |i,x,r \oplus \Pi^{-1}_{|i|}(x)\rangle & , \tilde{\Pi}^{-1}_{|i|}(x) = 0, G((i-1)N+\Pi^{-1}_{|i|}(x)) = 0, i < 0 \\ |i,x,r \oplus \bar{x}_i\rangle & , \tilde{\Pi}^{-1}_{|i|}(x) = 0, G((i-1)N+\Pi^{-1}_{|i|}(x)) = 1, i < 0. \end{cases} \tag{13}$$

By Lemma 20, if we try to distinguish f from g by distinguishing $\mathcal{A}^{S\Pi^G}$ of the two cases, we can only succeed with probability $O\left(\ell\sqrt{\frac{k}{N}}\right)$ since we only have $O(\ell)$ k-parallel queries to G. Therefore, one can only distinguish $S\Pi$ from $S\tilde{\Pi}'$ with probability $O\left(\ell\sqrt{\frac{k}{N}}\right)$, i.e.,

$$\Delta\left(\rho_{S\Pi}, \mathop{\mathbb{E}}_{S\tilde{\Pi}' \in H^{-1}(S\Pi)}[\rho_{S\tilde{\Pi}'}]\right) = O\left(\ell\sqrt{\frac{k}{N}}\right). \tag{14}$$

Then by (12),

$$\Delta \left(\underset{\mathrm{S\tilde{\Pi}} \in F^{-1}(\mathrm{S\tilde{\Pi}})}{\mathbb{E}} [\rho_{\mathrm{S\Pi}}], \underset{\mathrm{S\tilde{\Pi}'} \in H^{-1}(F^{-1}(\mathrm{S\tilde{\Pi}}))}{\mathbb{E}} [\rho_{\mathrm{S\tilde{\Pi}'}}] \right) = O\left(\ell \sqrt{\frac{k}{N}} \right). \tag{15}$$

By the operational interpretation of the trace distance, we have

$$\underset{\mathrm{S\Pi} \in F^{-1}(\mathrm{S\tilde{\Pi}})}{\mathbb{E}} \left[\Pr[\mathcal{A}^{\mathrm{S\Pi}} \neq \tilde{\mathcal{A}}^{\mathrm{S\tilde{\Pi}}}] \right] = O\left(\ell \sqrt{\frac{k}{N}} \right). \tag{16}$$

◄

▶ **Lemma 22.** *Use the notations of Definition 17. Let ℓ, k be integers such that $\ell \in [q]$, $k = O(\mathrm{polylog}(N))$. For all algorithm $\mathcal{A}$ using ℓ k-parallel queries to $\mathrm{S\tilde{\Pi}}$, w.l.o.g. we can assume the final output of $\mathcal{A}$ has the form*

$$|\psi\rangle = U_\ell \mathrm{S\tilde{\Pi}}^{\otimes k} U_{\ell-1} \mathrm{S\tilde{\Pi}}^{\otimes k} \dots U_2 \mathrm{S\tilde{\Pi}}^{\otimes k} U_1 \mathrm{S\tilde{\Pi}}^{\otimes k} |\psi_0\rangle.$$

For all $m, p \in [\ell]$, $p \leq m$, define the hybrid state

$$|\psi_{m,p}\rangle := U_m \mathrm{S\tilde{\Pi}}^{\otimes k} U_{m-1} \mathrm{S\tilde{\Pi}}^{\otimes k} \dots U_{p+1} \mathrm{S\tilde{\Pi}}^{\otimes k} U_p \mathrm{S\tilde{\Pi}}_p^{\otimes k} U_{p-1} \mathrm{S\tilde{\Pi}}_{p-1}^{\otimes k} \dots U_2 \mathrm{S\tilde{\Pi}}_2^{\otimes k} U_1 \mathrm{S\tilde{\Pi}}_1^{\otimes k} |\psi_0\rangle.$$

Then for all $\mathcal{A}$,

$$\underset{\mathrm{S\tilde{\Pi}}}{\mathbb{E}} \| |\psi\rangle - |\psi_{\ell,\ell}\rangle \| \leq 2\ell \sqrt{\frac{k}{N}}. \tag{17}$$

Proof. Note that $|\psi\rangle = |\psi_{\ell,0}\rangle$. By triangle inequality we have

$$\underset{\mathrm{S\tilde{\Pi}}}{\mathbb{E}} \| |\psi\rangle - |\psi_{\ell,\ell}\rangle \| \leq \sum_{i=1}^{\ell} \underset{\mathrm{S\tilde{\Pi}}}{\mathbb{E}} \| |\psi_{\ell,i-1}\rangle - |\psi_{\ell,i}\rangle \|. \tag{18}$$

We proceed by proving $\mathbb{E}_{\mathrm{S\tilde{\Pi}}} \| |\psi_{\ell,i-1}\rangle - |\psi_{\ell,i}\rangle \| = 2\sqrt{k/N}$ for all $i \in [\ell]$. Note that

$$\begin{aligned} \| |\psi_{\ell,i-1}\rangle - |\psi_{\ell,i}\rangle \| &= \| |\psi_{i,i-1}\rangle - |\psi_{i,i}\rangle \| \\ &= \| U_i \mathrm{S\tilde{\Pi}}^{\otimes k} |\psi_{i-1,i-1}\rangle - U_i \mathrm{S\tilde{\Pi}}_i^{\otimes k} |\psi_{i-1,i-1}\rangle \| \\ &= \| (\mathrm{S\tilde{\Pi}}^{\otimes k} - \mathrm{S\tilde{\Pi}}_i^{\otimes k}) |\psi_{i-1,i-1}\rangle \|. \end{aligned} \tag{19}$$

Note that $\mathrm{S\tilde{\Pi}}|j, x, r\rangle$ and $\mathrm{S\tilde{\Pi}}_i|j, x, r\rangle$ differs only when $j > i$ and $x = \bar{x}_j$ or $j < -i$ and $x = \bar{x}_{j+1}$. Therefore

$$(\mathrm{S\tilde{\Pi}} - \mathrm{S\tilde{\Pi}}_i)|j, x, r\rangle = (\mathrm{S\tilde{\Pi}} - \mathrm{S\tilde{\Pi}}_i)P_i|j, x, r\rangle \tag{20}$$

$$\mathrm{S\tilde{\Pi}}(1 - P_i) = \mathrm{S\tilde{\Pi}}_i(I - P_i) \tag{21}$$

$$(\mathrm{S\tilde{\Pi}}^{\otimes k} - \mathrm{S\tilde{\Pi}}_i^{\otimes k})(I - P_i)^{\otimes k} = 0 \tag{22}$$

where

$$P_i := \left(\sum_{j=i+1}^{k} |j, \bar{x}_j\rangle \langle j, \bar{x}_j| + \sum_{-j=i+1}^{k} |j, \bar{x}_{j+1}\rangle \langle j, \bar{x}_{j+1}| \right) \otimes I. \tag{23}$$

Note that P_i actually depends on $\mathrm{S\tilde{\Pi}}$, but we omit the dependence for cleaner notation. Therefore we have

$$\left\|(S\tilde{\Pi}^{\otimes k} - S\tilde{\Pi}_i^{\otimes k})|\psi_{i-1,i-1}\rangle\right\|^2$$
$$= \left\|(S\tilde{\Pi}^{\otimes k} - S\tilde{\Pi}_i^{\otimes k})(I - (I - P_i)^{\otimes k})|\psi_{i-1,i-1}\rangle\right\|^2$$
$$\leq 4\left\|(I - (I - P_i)^{\otimes k})|\psi_{i-1,i-1}\rangle\right\|^2$$
$$= 4\langle\psi_{i-1,i-1}|(I - (I - P_i)^{\otimes k})|\psi_{i-1,i-1}\rangle$$
$$\leq 4\langle\psi_{i-1,i-1}|\sum_{j=1}^{k} P_i^j|\psi_{i-1,i-1}\rangle \tag{24}$$

where $P_i^j = I^{\otimes j-1} \otimes P_i \otimes I^{\otimes k-j}$. In the fourth line, we use the fact that $I - (I - P_i)^{\otimes k}$ is a projector, and in the fifth line we use the standard union bound calculation.

By (23), for all $i \in [q]$ and $j \in [k]$, P_i^j is normalized by

$$\sum_{\Delta\bar{x}_i=0}^{N-1} P_i^j = \left(\sum_{j=i+1}^{k} + \sum_{-j=i+1}^{k}\right)|j\rangle\langle j| \otimes \sum_{\bar{x}\in[N]}|\bar{x}\rangle\langle\bar{x}| \leq I \tag{25}$$

where we omitted the tensor product of identities and assume $\Delta\bar{x}_{i+1}, \ldots, \Delta\bar{x}_q$ to be fixed. Therefore

$$\sum_{\Delta\bar{x}_i=0}^{N-1} \left\|(S\tilde{\Pi}^{\otimes k} - S\tilde{\Pi}_i^{\otimes k})|\psi_{i-1,i-1}\rangle\right\|^2$$
$$\leq \sum_{\Delta\bar{x}_i=0}^{N-1} 4\langle\psi_{i-1,i-1}|\sum_{j=1}^{k} P_i^j|\psi_{i-1,i-1}\rangle$$
$$\leq 4k \tag{26}$$

Finally, combining (26), (24), and (19), we have

$$\mathop{\mathbb{E}}_{S\tilde{\Pi}}\left\||\psi_{\ell,i-1}\rangle - |\psi_{\ell,i}\rangle\right\|$$
$$= \mathop{\mathbb{E}}_{\Delta\bar{x}_1,\ldots,\Delta\bar{x}_{i-1}} \mathop{\mathbb{E}}_{\Delta\bar{x}_i} \mathop{\mathbb{E}}_{\Delta\bar{x}_{i+1},\ldots,\Delta\bar{x}_q} \left\|(S\tilde{\Pi}^{\otimes k} - S\tilde{\Pi}_i^{\otimes k})|\psi_{i-1,i-1}\rangle\right\|$$
$$= \mathop{\mathbb{E}}_{\Delta\bar{x}_1,\ldots,\Delta\bar{x}_{i-1}} \mathop{\mathbb{E}}_{\Delta\bar{x}_{i+1},\ldots,\Delta\bar{x}_q} \frac{1}{N}\sum_{\Delta\bar{x}_i} \left\|(S\tilde{\Pi}^{\otimes k} - S\tilde{\Pi}_i^{\otimes k})|\psi_{i-1,i-1}\rangle\right\|$$
$$\leq \mathop{\mathbb{E}}_{\Delta\bar{x}_1,\ldots,\Delta\bar{x}_{i-1}} \mathop{\mathbb{E}}_{\Delta\bar{x}_{i+1},\ldots,\Delta\bar{x}_q} \frac{1}{N}\sqrt{\sum_{\Delta\bar{x}_i}\left\|(S\tilde{\Pi}^{\otimes k} - S\tilde{\Pi}_i^{\otimes k})|\psi_{i-1,i-1}\rangle\right\|^2} \cdot \sqrt{N}$$
$$= 2\sqrt{\frac{k}{N}}. \tag{27}$$

Plugging back to (18) we have

$$\mathop{\mathbb{E}}_{S\tilde{\Pi}}\left\||\psi\rangle - |\psi_{\ell,\ell}\rangle\right\| \leq 2\ell\sqrt{\frac{k}{N}}. \tag{28}$$

◀

▶ **Corollary 23.** *Any algorithm $\mathcal{A}$ using $(q-1)$ k-parallel queries to $S\tilde{\Pi}$ can only output $\bar{x}_{q+1}$ with probability $O\left(\ell\sqrt{\frac{k}{N}}\right)$.*

Proof. By Lemma 19 and Lemma 22, output probability of $|\psi\rangle$ and $|\psi_{\ell,\ell}\rangle$ only differ by $O(\ell\sqrt{k/N})$. Since $|\psi_{q-1,q-1}\rangle$ does not depend on $\tilde{\Pi}_q$, it can only find $\bar{x}^{q+1}$ with probability $1/N$. Thus the maximum probability of $|\psi\rangle$ finding $\bar{x}^{q+1}$ is $O(\ell\sqrt{k/N}) + 1/N = O(\ell\sqrt{k/N})$.

◀

Proof of Theorem 18. Let $\mathcal{A}$ be an algorithm that uses $\lfloor (q-1)/2 \rfloor$ k-parallel queries to SΠ and outputs $\bar{x}_{q+1}$ with some probability p. By Lemma 21, there is an algorithm $\tilde{\mathcal{A}}$ that uses $(q-1)$ k-parallel queries to SΠ̃ and outputs $\bar{x}_{q+1}$ with probability at least $p - O\left(q\sqrt{\frac{k}{N}}\right)$. By Corollary 23, it holds that $p - O\left(q\sqrt{\frac{k}{N}}\right) = O\left(q\sqrt{\frac{k}{N}}\right)$. Therefore, we have $p = O\left(q\sqrt{\frac{k}{N}}\right)$.

◀

6   Parallel Hardness of Twisted Hash Chains

The (standard) hash chain problem is a natural candidate for parallel hardness. However, hash functions (modeled as random functions) with equal-length inputs and outputs are not injective with overwhelming probability. Importantly, quantum operations need to be reversible. Therefore, if we evaluate the standard hash chain straight-forwardly, the intermediate hash values are required to be stored in ancilla qubits. Consequently, the width of the circuit would become proportional to the length of the chain, which means the computation could not be done within width λ.

Inspired by the construction of the Feistel cipher which implements a permutation by hash functions, we introduce the *twisted hash chain* problem. Let $\mathcal{X}$ be $\{0,1\}^n$. An *s-chain* is a sequence $x_0, x_1, \ldots, x_s \in \mathcal{X}$ such that $x_i = H(x_{i-1}) \oplus x_{i-2}$ for $i \in [s]$ where we use the convention that $x_{-1} := 0^n$. Informally, the task of a q-query k-parallel algorithm that interacts with a random oracle $H : \mathcal{X} \to \mathcal{Y}$ with $|\mathcal{X}| = |\mathcal{Y}|$ is to output $x_0, x_q, x_{q+1} \in \mathcal{X}$ of a $(q+1)$-chain, i.e., there exists a sequence $x_1, \ldots, x_{q-1} \in \mathcal{X}$ such that $x_i = H(x_{i-1}) \oplus x_{i-2}$ for $i \in [q+1]$. The computation of a twisted hash chain is shown in Fig. 3.
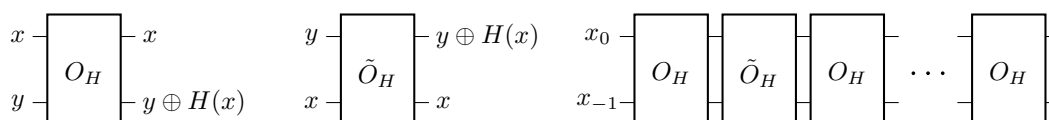
Figure 3 Schematic diagram of the twisted hash chain.

In this section, we aim to prove the following theorem.

▶ **Theorem 24** (Twisted hash chain is sequential). *For any k-parallel q-query oracle algorithm $\mathcal{C}$, the probability $p_\mathcal{C}$ (parameterized by k and q) that $\mathcal{C}$ outputs $x_0, x_q, x_{q+1} \in \mathcal{X}$ satisfying the following condition:*
▪ *there exist $x_1, \ldots, x_{q-1} \in \mathcal{X}$ such that $H(x_{i-1}) = x_i \oplus x_{i-2}$ for $i \in [q+1]$, where $x_{-1} := 0^n$*
is at most $F(k, 2q) = O(k^4 q^4 / |\mathcal{Y}|)$, where the function F is defined in Lemma 32.

Toward proving the hardness, we exploit the framework of [16]. Below, we borrow the notations and definitions of [16]. Let $H : \mathcal{X} \to \mathcal{Y}$ be a random oracle. Let $\hat{\mathcal{Y}}$ be the dual group of $\mathcal{Y}$. Let $\overline{\mathcal{Y}}$ denote the set $\mathcal{Y} \cup \{\bot\}$. We say that $D : \mathcal{X} \to \overline{\mathcal{Y}}$ is a *database*. By $\mathfrak{D}$ we mean the set of all databases, i.e., the set of all functions from $\mathcal{X}$ to $\overline{\mathcal{Y}}$. For any tuple $\mathbf{x} = (x_1, \ldots, x_k)$ with pairwise disjoint $x_i \in \mathcal{X}$, tuple $\mathbf{r} = (r_1, \ldots, r_k) \in \overline{\mathcal{Y}}^k$ and database $D \in \mathfrak{D}$, we define the database $D[\mathbf{x} \mapsto \mathbf{r}]$ as

$$D[\mathbf{x} \mapsto \mathbf{r}](x) := \begin{cases} r_i & \text{if } x = x_i \text{ for some } i \in [k] \\ D(x) & \text{if } x \notin \{x_1, \ldots, x_k\}. \end{cases}$$

By database property P we mean a set of databases, that is $P \subseteq \mathfrak{D}$. In this section, we assume that $\mathcal{X} = \mathcal{Y}$. For any database $D \in \mathfrak{D}$ and tuple $\mathbf{x} = (x_1, \ldots, x_k)$ of pairwise distinct $x_i \in \mathcal{X}$, we let

$$D|^{\mathbf{x}} := \{D[\mathbf{x} \mapsto \mathbf{r}] \mid \mathbf{r} \in \overline{\mathcal{Y}}^k\} \subseteq \mathfrak{D}$$

be the set of databases that coincide D outside of $\mathbf{x}$. Furthermore, for any database property $\mathsf{P} \subseteq \mathfrak{D}$, we let

$$\mathsf{P}|_{D|^{\mathbf{x}}} := \mathsf{P} \cap D|^{\mathbf{x}}.$$

▶ **Definition 25** (Definition 5.5 in [16]). *Let* P, P' *be two database properties. Then, the quantum transition capacity (of order k) is defined as*

$$\llbracket \mathsf{P} \xrightarrow{k} \mathsf{P}' \rrbracket := \max_{\mathbf{x}, \hat{\mathbf{y}}, D} \|\mathsf{P}'|_{D|^{\mathbf{x}}} \mathsf{cO}_{\mathbf{x}\hat{\mathbf{y}}} \mathsf{P}|_{D|^{\mathbf{x}}}\|,$$

where the maximum is over all possible $\mathbf{x} \in \mathcal{X}^k$, $\hat{\mathbf{y}} \in \hat{\mathcal{Y}}^k$ *and* $D \in \mathfrak{D}$. *Furthermore, we define*

$$\llbracket \mathsf{P} \xRightarrow{k} \mathsf{P}' \rrbracket := \sup_{U_1, \ldots, U_{q-1}} \left\| \mathsf{P}' U_{q-1} \mathsf{cO}^k U_{q-1} \mathsf{cO}^k \ldots U_1 \mathsf{cO}^k \mathsf{P} \right\|,$$

where $\| \cdot \|$ *is the operator norm; the supremum is over all positive* $d \in \mathbb{Z}$ *and all unitaries* $U_1, \ldots, U_{q-1}$ *acting on* $\mathbb{C}[\mathcal{X}] \otimes \mathbb{C}[\mathcal{Y}] \otimes \mathbb{C}^d$.[12] *For the formal definitions of* $\mathsf{cO}_{\mathbf{x}\hat{\mathbf{y}}}$ *and* cO^k, *we refer to [16]. We note that their definitions are not required for the following proof.*

▶ **Definition 26.** *The database property* twisted hash chain of length s, *denoted by* TCHN^s, *is defined as*

$$\mathsf{TCHN}^s := \{D \mid \exists x_0, x_1, \ldots, x_s \in \mathcal{X} \colon x_i = D(x_{i-1}) \oplus x_{i-2}, \forall i \in [s]\} \subseteq \mathfrak{D},$$

where we use the convention that $x_{-1} := 0^n$ *for convenience.*

▶ **Definition 27** (Definition 5.20 in [16], with ℓ fixed to 1). *A database transition* $\mathsf{P} \to \mathsf{P}'$ *is said be k-non-uniformly weakly recognizable by 1-local properties[13] if for every* $\mathbf{x} = (x_1, \ldots, x_k)$ *with pairwise disjoint entries, and for every* $D \in \mathfrak{D}$, *there exists a family of 1-local properties* $\{L_i^{D,\mathbf{x}}\}$ *where each* $L_i^{D,\mathbf{x}} \subseteq \overline{\mathcal{Y}}$ *and the support of* $L_i^{D,\mathbf{x}}$ *is* $\{x_i\}$ *or empty, so that*

$$D[\mathbf{x} \mapsto \mathbf{r}] \in \mathsf{P} \wedge [\mathbf{x} \mapsto \mathbf{u}] \in \mathsf{P}' \implies \exists i \colon u_i \in L_i^{D,\mathbf{x}} \wedge r_i \neq u_i.$$

▶ **Theorem 28** (Theorem 5.23 in [16]). *Let* P *and* P' *be k-non-uniformly weakly recognizable by 1-local properties* $L_i^{\mathbf{x},D}$, *where the support of* $L_i^{\mathbf{x},D}$ *is* $\{x_i\}$ *or empty. Then*

$$\llbracket \perp \xRightarrow{q,k} \mathsf{TCHN}^{q+1} \rrbracket \leq \max_{\mathbf{x},D} e \sum_i \sqrt{10P[U \in L_i^{\mathbf{x},D}]},$$

where U is defined to be uniformly random in $\mathcal{Y}$ *and* $\perp := \{D \mid D(x) = \perp \text{ for all } x \in \mathcal{X}\}$.

---

[12] Namely, over all q-query quantum algorithms.
[13] We refer to Definition 5.10 in [16] for the formal description of local properties.

Here, we define the family of local properties for our purpose. For any $D \in \mathfrak{D}$ and any $\mathbf{x} := (x_1, \dots, x_k) \in \mathcal{X}^k$ with disjoint entries, we defined the following 1-local properties $L_i^{D,\mathbf{x}} \subseteq \overline{\mathcal{Y}}$[14] with support $\{x_i\}$ for $i \in [k]$ as

$$L_i^{D,\mathbf{x}} := L_{i,1}^{D,\mathbf{x}} \cup L_{i,2}^{D,\mathbf{x}},$$

where

$$L_{i,1}^{D,\mathbf{x}} := \{x \in \mathcal{X} \mid D(x) \neq \bot \vee x \in \{x_1, \dots, x_k\}\}$$

and

$$L_{i,2}^{D,\mathbf{x}} := \{x \in \mathcal{X} \mid \exists x', x'' \in L_{i,1}^{D,\mathbf{x}} : x = x' \oplus x''\}.$$

The following lemma, in line with Lemma 2.1 in [16], shows that the local properties $\{L_i^{D,\mathbf{x}}\}$ recognize the database transition $\neg\mathsf{TCHN}^\mathsf{s} \to \mathsf{TCHN}^{\mathsf{s}+1}$, and allows us to exploit Theorem 28. First, we briefly explain the intuition. We pick an arbitrary $(s+1)$-chain in $D[\mathbf{x} \mapsto \mathbf{u}]$ and call it the *new chain*. We denote the elements of the new chain by $\hat{x}_0, \hat{x}_1, \dots, \hat{x}_{s+1}$. There are two possible consequences of this database transition:

First, some branch x_i of the query $\mathbf{x}$ becomes the first elements $\hat{x}_0$ of the new chain, i.e., $x_i = \hat{x}_0$ and x_i is responded with $D[\mathbf{x} \mapsto \mathbf{u}](x_i) = u_i$ such that $u_i = D[\mathbf{x} \mapsto \mathbf{u}](\hat{x}_0) = \hat{x}_1$. In addition, $D[\mathbf{x} \mapsto \mathbf{u}](\hat{x}_1) = \hat{x}_0 \oplus \hat{x}_2 \neq \bot$. This means that $\hat{x}_1$ must either already be sampled $(D(\hat{x}_1) \neq \bot)$ or be one of the branch of $\mathbf{x}$ $(\hat{x}_1 \in \{x_1, \dots, x_k\})$ (or both). In other words, u_i must be in $L_{i,1}^{D,\mathbf{x}}$.

Second, some branch x_i of the query $\mathbf{x}$ becomes the $(j+1)$-th elements $\hat{x}_j$ $(1 \leq j \leq s)$ of the new chain, i.e., $x_i = \hat{x}_j$ and x_i is responded with $D[\mathbf{x} \mapsto \mathbf{u}](x_i) = u_i$ such that $u_i = D[\mathbf{x} \mapsto \mathbf{u}](\hat{x}_j) = \hat{x}_{j-1} \oplus \hat{x}_{j+1}$. Similarly, we can conclude that either $D(\hat{x}_{j-1}) \neq \bot$ or $\hat{x}_{j-1} \in \{x_1, \dots, x_k\}$ (or both) and either $D(\hat{x}_{j+1}) \neq \bot$ or $\hat{x}_{j+1} \in \{x_1, \dots, x_k\}$ (or both). This means that u_i must be the XOR of two elements in $L_{i,1}^{D,\mathbf{x}}$. That is, $u_i \in L_{i,2}^{D,\mathbf{x}}$.

The intuition above can be formalized as the following lemma.

▶ **Lemma 29.** $D[\mathbf{x} \mapsto \mathbf{r}] \notin \mathsf{TCHN}^\mathsf{s} \wedge D[\mathbf{x} \mapsto \mathbf{u}] \in \mathsf{TCHN}^{\mathsf{s}+1} \implies \exists i \in [k] : r_i \neq u_i \wedge u_i \in L_i^{D,\mathbf{x}}$.

Proof. Suppose $D[\mathbf{x} \mapsto \mathbf{u}] \in \mathsf{TCHN}^{\mathsf{s}+1}$ and let $\hat{x}_0, \hat{x}_1, \dots, \hat{x}_{s+1}$ be such a chain, i.e., $\hat{x}_1 = D[\mathbf{x} \mapsto \mathbf{u}](\hat{x}_0)$ and $\hat{x}_{j+2} = D[\mathbf{x} \mapsto \mathbf{u}](\hat{x}_{j+1}) \oplus \hat{x}_j$ for $j = 0, 1, \dots, q-1$. Let $s_\circ$ be the smallest j such that $D[\mathbf{x} \mapsto \mathbf{r}](\hat{x}_{s_\circ}) \neq D[\mathbf{x} \mapsto \mathbf{u}](\hat{x}_{s_\circ})$. If $s_\circ \geq s$ or j does not exist, then $D[\mathbf{x} \mapsto \mathbf{r}] \in \mathsf{TCHN}^{\mathsf{s}+1}$, and we are done. Suppose now $0 \leq s_\circ \leq s-1$. Since $D[\mathbf{x} \mapsto \mathbf{u}]$ and $D[\mathbf{x} \mapsto \mathbf{u}]$ are identical outside of $\mathbf{x}$, there exists a coordinate i of $\mathbf{x}$ such that $x_i = \hat{x}_{s_\circ}$. Therefore, $u_i = D[\mathbf{x} \mapsto \mathbf{u}](x_i) = D[\mathbf{x} \mapsto \mathbf{u}](\hat{x}_{s_\circ}) \neq D[\mathbf{x} \mapsto \mathbf{r}](\hat{x}_{s_\circ}) = D[\mathbf{x} \mapsto \mathbf{r}](x_i) = r_i$.

Below, we divide the analysis into three cases according to the value of $s_\circ$:

1. If $s_\circ = 0$, we have $u_i = D[\mathbf{x} \mapsto \mathbf{u}](\hat{x}_{s_\circ}) = D[\mathbf{x} \mapsto \mathbf{u}](\hat{x}_0) = \hat{x}_1$. And $D[\mathbf{x} \mapsto \mathbf{u}](\hat{x}_1) = \hat{x}_2 \oplus \hat{x}_0 \neq \bot$, which implies either $D(\hat{x}_1) \neq \bot$ or $\hat{x}_1 \in \{x_1, \dots, x_k\}$ (or both). This means $u_i \in L_{i,1}^{D,\mathbf{x}}$.

2. If $s_\circ = 1$, we have $u_i = D[\mathbf{x} \mapsto \mathbf{u}](\hat{x}_{s_\circ}) = D[\mathbf{x} \mapsto \mathbf{u}](\hat{x}_1) = \hat{x}_2 \oplus \hat{x}_0$. And $D[\mathbf{x} \mapsto \mathbf{u}](\hat{x}_2) = \hat{x}_3 \oplus \hat{x}_1 \neq \bot$, which implies either $D(\hat{x}_2) \neq \bot$ or $\hat{x}_2 \in \{x_1, \dots, x_k\}$ (or both). And $D[\mathbf{x} \mapsto \mathbf{u}](\hat{x}_0) = \hat{x}_1 \neq \bot$, which implies either $D(\hat{x}_0) \neq \bot$ or $\hat{x}_0 \in \{x_1, \dots, x_k\}$ (or both). This means $u_i \in L_{i,2}^{D,\mathbf{x}}$.

---

[14] Recall that we assume $\mathcal{X} = \mathcal{Y}$.

3. If $2 \leq s_\circ \leq s - 1$, then $r_i = D[\mathbf{x} \mapsto \mathbf{u}](\hat{x}_{s_\circ}) = \hat{x}_{s_\circ+1} \oplus \hat{x}_{s_\circ-1}$. Similarly, $D[\mathbf{x} \mapsto \mathbf{u}](\hat{x}_{s_\circ+1}) = \hat{x}_{s_\circ} \oplus \hat{x}_{s_\circ+2} \neq \bot$, which implies either $D(\hat{x}_{s_\circ+1}) \neq \bot$ or $\hat{x}_{s_\circ+1} \in \{x_1, \ldots, x_k\}$ (or both). And $D[\mathbf{x} \mapsto \mathbf{u}](\hat{x}_{s_\circ-1}) = \hat{x}_{s_\circ} \oplus \hat{x}_{s_\circ-2} \neq \bot$, which implies either $D(\hat{x}_1) \neq \bot$ or $\hat{x}_1 \in \{x_1, \ldots, x_k\}$ (or both). This means $u_i \in L_{i,2}^{D,\mathbf{x}}$.

In all of the above cases, u_i must be in $L_i^{D,\mathbf{x}}$ which concludes the proof. ◄

We need Corollary 4.2 in [16] which is rephrased from Lemma 5 in [38].

▶ **Lemma 30** (Lemma 5 in [38]). *Let $R \subseteq \mathcal{X}^\ell \times \mathcal{Y}^\ell$ be a relation. Let $\mathcal{A}$ be an algorithm that outputs $\mathbf{x} \in \mathcal{X}^\ell$ and $\mathbf{y} \in \mathcal{Y}^\ell$. Let p be the probability that $\mathbf{y} = H(\mathbf{x}) := (H(x_1), \ldots, H(x_\ell))$ and $(\mathbf{x}, \mathbf{y}) \in R$ when $\mathcal{A}$ has interacted with the standard random oracle, initialized with a uniformly random function H. Similarly, let p' be the probability that $\mathbf{y} = D(\mathbf{x}) := (D(x_1), \ldots, D(x_\ell))$ and $(\mathbf{x}, \mathbf{y}) \in R$ when $\mathcal{A}$ has interacted with the compressed oracle and D is obtained by measuring its internal state in the computational basis. Then*

$$\sqrt{p} \leq \sqrt{p'} + \sqrt{\frac{\ell}{|\mathcal{Y}|}}.$$

▶ **Lemma 31.** $\llbracket \bot \xrightarrow{q,k} \mathsf{TCHN}^{q+1} \rrbracket \leq qek\sqrt{\frac{5kq(kq+1)}{|\mathcal{Y}|}}.$

Proof. By Lemma 5.6 in [16], we have

$$\llbracket \bot \xrightarrow{q,k} \mathsf{TCHN}^{q+1} \rrbracket \leq \sum_{s=1}^{q} \llbracket \mathsf{SZ}_{\leq k(s-1)} \setminus \mathsf{TCHN}^s \xrightarrow{k} \mathsf{TCHN}^{s+1} \rrbracket.$$

Choosing the local properties $\{L_i^{D,\mathbf{x}}\}$ as above whenever $D \in \mathsf{SZ}_{\leq k(s-1)}$, and to be constant-false otherwise, Lemma 29 ensures that we can apply Theorem 5.23 in [16] to bound quantum transition capacity. Therefore, applying Theorem 5.23 in [16], for each $s \in [q]$ we have

$$\llbracket \mathsf{SZ}_{\leq k(s-1)} \setminus \mathsf{TCHN}^s \xrightarrow{k} \mathsf{TCHN}^{s+1} \rrbracket \leq e \max_{D,\mathbf{x}} \sum_{i=1}^{k} \sqrt{10 \Pr[U \in L_i^{D,\mathbf{x}}]} \leq ek\sqrt{\frac{5kq(kq+1)}{|\mathcal{Y}|}}.$$

The last inequality holds because for every $s \in [q]$ and every $D \in \mathsf{SZ}_{\leq k(s-1)}$, it holds that

$$|\{x \in \mathcal{X} \mid D(x) \neq \bot\} \cup \{x_1 \ldots, x_k\}| \leq k(q-1) + k = kq.$$

Thus, we have $|L_{i,1}^{D,\mathbf{x}}| \leq kq$ and $|L_{i,2}^{D,\mathbf{x}}| \leq \binom{kq}{2} = kq(kq-1)/2$ for $i \in [k]$, which then implies $|L_i^{D,\mathbf{x}}| \leq kq(kq+1)/2$. Finally, summing over $s \in [q]$ completes the proof. ◄

First, note that Lemma 30 is tailored for algorithms that output *all elements of the chain and their hash values*. However, to obtain a bound when the algorithm $\mathcal{A}$ is required to output *only x_0, x_q and x_{q+1}* is more challenging. In most of situations, one could define another algorithm $\mathcal{B}$ that simply runs $\mathcal{A}$ followed by calculating the whole chain with $q + 2$ extra queries. This would increase the number of queries by at most $q + 2$. However, this gives us a meaningless bound for the twisted hash chain problem.

Below, we first provide the following lemma for algorithms that do *not* have to output the last hash value y_{q+1}. The proof is similar to Theorem 5.9 in [16].

▶ **Lemma 32.** *For any k-parallel q-query oracle algorithm $\mathcal{A}$ that interacts with a standard random oracle, the probability $p_{\mathcal{A}}$ (parameterized by k and q) that $\mathcal{A}$ outputs $x_0, x_1, \ldots, x_{q+1} \in \mathcal{X}$ and $y_0, y_1, \ldots, y_q \in \mathcal{Y}$ (without y_{q+1}) satisfying*

- $y_i = H(x_i)$ *for* $0 \leq i \leq q$
- $y_{i-1} = x_i \oplus x_{i-2}$ *for* $i \in [q+1]$, *where* $x_{-1} := 0^n$

is upper bounded by the function $F(k, q)$ *where*

$$F(k, q) := \left(qek\sqrt{\frac{5kq(kq+1)}{|\mathcal{Y}|}} + e(q+2)\sqrt{\frac{5(q+2)(q+3)}{|\mathcal{Y}|}} + \sqrt{\frac{q+2}{|\mathcal{Y}|}} \right)^2 = O\left(\frac{q^4 k^4}{|\mathcal{Y}|} \right).$$

Proof. We define an algorithm $\mathcal{B}$ that runs $\mathcal{A}$ and obtains the output $x_0, \dot{,} x_{q+1}, y_0, \ldots, y_q$. And then $\mathcal{B}$ makes a classical query x_{q+1} to the random oracle. Finally, $\mathcal{B}$ outputs $x_0, \dot{,} x_{q+1}, y_0, \ldots, y_q, H(x_{q+1})$.

Let $p_{\mathcal{B}}$ be the probability that $\mathcal{B}$ outputs $x_0, x_1, \ldots, x_{q+1} \in \mathcal{X}$ and $y_0, y_1, \ldots, y_{q+1} \in \mathcal{Y}$ satisfying

- $y_i = H(x_i)$ for $0 \leq i \leq q+1$
- $y_{i-1} = x_i \oplus x_{i-2}$ for $i \in [q+1]$, where $x_{-1} := 0^n$

when $\mathcal{B}$ interacts with the standard random oracle.

Let $p'_{\mathcal{B}}$ be the probability that $\mathcal{B}$ outputs $x_0, x_1, \ldots, x_{q+1} \in \mathcal{X}$ and $y_0, y_1, \ldots, y_{q+1} \in \mathcal{Y}$ satisfying

- $y_i = D(x_i)$ for $0 \leq i \leq q+1$
- $y_{i-1} = x_i \oplus x_{i-2}$ for $i \in [q+1]$, where $x_{-1} := 0^n$

when $\mathcal{B}$ interacts with the compressed oracle.

Let $\overline{p}'_{\mathcal{B}}$ be the probability that $\mathcal{B}$ outputs $x_0, x_1, \ldots, x_{q+1} \in \mathcal{X}$ and $y_0, y_1, \ldots, y_{q+1} \in \mathcal{Y}$ satisfying

- $D(x_{i-1}) = x_i \oplus x_{i-2}$ for $i \in [q+1]$, where $x_{-1} := 0^n$

when $\mathcal{B}$ interacts with the compressed oracle.

We trivially have $p_{\mathcal{A}} = p_{\mathcal{B}}$ and $p'_{\mathcal{B}} \leq \overline{p}'_{\mathcal{B}}$. Since $\mathcal{B}$ now outputs all the hash values as well, we can apply Lemma 30 to $\mathcal{B}$ which gives

$$\sqrt{p_{\mathcal{B}}} \leq \sqrt{p'_{\mathcal{B}}} + \sqrt{\frac{q+2}{|\mathcal{Y}|}}.$$

In the rest of the proof, it remains to bound $\overline{p}'_{\mathcal{B}}$.

$$\sqrt{\overline{p}'_{\mathcal{B}}} \leq \sup_{U_1, \ldots, U_q} \left\| \sum_{\mathbf{x}} \mathsf{TCHN}_{\mathbf{x}}^{q+1}(|\mathbf{x}\rangle\langle\mathbf{x}| \otimes \mathsf{cO}_{x_{q+1}}) U_q \mathsf{cO}^k U_{q-1} \mathsf{cO}^k \ldots U_1 \mathsf{cO}^k \perp \right\|$$

$$\leq \left\| \sum_{\mathbf{x}} \mathsf{TCHN}_{\mathbf{x}}^{q+1}(|\mathbf{x}\rangle\langle\mathbf{x}| \otimes \mathsf{cO}_{x_{q+1}}) \neg \mathsf{TCHN}^{q+1} \right\|$$

$$+ \sup_{U_1, \ldots, U_q} \| \mathsf{TCHN}^{q+1} U_q \mathsf{cO}^k U_{q-1} \mathsf{cO}^k \ldots U_1 \mathsf{cO}^k \perp \|$$

$$\leq \max_{\mathbf{x}} \| \mathsf{TCHN}_{\mathbf{x}}^{q+1} \mathsf{cO}_{x_{q+1}} \neg \mathsf{TCHN}^{q+1} \| + [\![\perp \xRightarrow{q,k} \mathsf{TCHN}^{q+1}]\!]$$

$$\leq \max_{\mathbf{x}} \| \mathsf{TCHN}_{\mathbf{x}}^{q+1} \mathsf{cO}_{x_{q+1}} \neg \mathsf{TCHN}_{\mathbf{x}}^{q+1} \| + [\![\perp \xRightarrow{q,k} \mathsf{TCHN}^{q+1}]\!],$$

where the summation is over all $\mathbf{x} = (x_0, \ldots, x_{q+1}) \in \mathcal{X}^{q+2}$; $\{|\mathbf{x}\rangle\langle\mathbf{x}|\}$ denotes the measurement acting on $\mathcal{B}$'s output register to produce the output $\mathbf{x}$; the database property $\mathsf{TCHN}_{\mathbf{x}}^{q+1}$ is defined as

$$\mathsf{TCHN}_{\mathbf{x}}^{q+1} := \{D \mid x_i = D(x_{i-1}) \oplus x_{i-2} \text{ for } i \in [q+1]\} \subseteq \mathfrak{D}.$$

That is, the sequence $x_0, \ldots, x_{q+1}$ forms a $(q+1)$-chain.

Now, notice that for every $\mathbf{x} \in \mathcal{X}^{q+2}$,

$$\|\mathsf{TCHN}_{\mathbf{x}}^{q+1}\mathsf{cO}_{x_{q+1}}\neg\mathsf{TCHN}_{\mathbf{x}}^{q+1}\|$$
$$=\|\mathsf{TCHN}_{\mathbf{x}}^{q+1}(\mathsf{cO}_{x_{q+1}} \otimes \mathsf{cO}_{x_q\hat{0}} \otimes \cdots \otimes \mathsf{cO}_{x_0\hat{0}})\neg\mathsf{TCHN}_{\mathbf{x}}^{q+1}\|$$
$$\leq \max_{\hat{\mathbf{y}}} \|\mathsf{TCHN}_{\mathbf{x}}^{q+1}\mathsf{cO}_{\mathbf{x}\hat{\mathbf{y}}}\neg\mathsf{TCHN}_{\mathbf{x}}^{q+1}\| \leq [\![\neg\mathsf{TCHN}_{\mathbf{x}}^{q+1} \xrightarrow{q+2} \mathsf{TCHN}_{\mathbf{x}}^{q+1}]\!],$$

where the first equality holds since $\mathsf{cO}_{x\hat{0}}$ is equal to the identity operator for every $x \in \mathcal{X}$.

Following similar arguments as in Lemma 29, we now show there exist local properties that recognize the database transition $[\![\neg\mathsf{TCHN}_{\mathbf{x}}^{q+1} \xrightarrow{q+2} \mathsf{TCHN}_{\mathbf{x}}^{q+1}]\!]$. For any tuple $\mathbf{x} = (x_1, \ldots, x_{q+2})$ with pairwise distinct entries, any tuple $\mathbf{x}' = (x_0', \ldots, x_{q+1}')$[15] and database $D \in \mathfrak{D}$, we define the following local properties for $i \in [q+2]$

$$L_i^{\mathbf{x},D} := \{x_0', \ldots, x_{q+1}'\} \cup \{x \mid \exists a, b \in \{1, \ldots, q+2\} \colon x = x_a' \oplus x_b'\}.$$

Note that $|L_i^{\mathbf{x},D}| \leq (q+2) + \binom{q+2}{2} = (q+2)(q+3)/2$ for each $i \in [q+2]$.

Suppose $D[\mathbf{x} \mapsto \mathbf{r}] \notin \mathsf{TCHN}_{\mathbf{x}'}^{q+1}$ yet $D[\mathbf{x} \mapsto \mathbf{u}] \in \mathsf{TCHN}_{\mathbf{x}'}^{q+1}$. Then $\{x_0', \ldots, x_{q+1}'\}$ is a $(q+1)$-chain. Let $s_\circ$ be the smallest j such that $D[\mathbf{x} \mapsto \mathbf{r}](x_j') \neq D[\mathbf{x} \mapsto \mathbf{u}](x_j')$. If $s_\circ = q+1$ or j does not exist, then $D[\mathbf{x} \mapsto \mathbf{r}] \in \mathsf{TCHN}_{\mathbf{x}'}^{q+1}$ and we are done. So we assume $0 \leq s_\circ \leq q$. Since $D[\mathbf{x} \mapsto \mathbf{r}]$ coincides $D[\mathbf{x} \mapsto \mathbf{u}]$ outside of $\mathbf{x}$, there must exists an index $i \in [q+2]$ such that $x_i = x_{s_\circ}'$. Therefore, we have $r_i = D[\mathbf{x} \mapsto \mathbf{r}](x_i) = D[\mathbf{x} \mapsto \mathbf{r}](x_{s_\circ}') \neq D[\mathbf{x} \mapsto \mathbf{u}](x_{s_\circ}') = D[\mathbf{x} \mapsto \mathbf{u}](x_i) = u_i$.

In addition, if $s_\circ = 0$, then $u_i = D[\mathbf{x} \mapsto \mathbf{u}](x_0') = x_1' \in \{x_0', \ldots, x_{q+1}'\}$. If $1 \leq s_\circ \leq q$, then $u_i = D[\mathbf{x} \mapsto \mathbf{u}](x_{s_\circ}') = x_{s_\circ-1}' \oplus x_{s_\circ+1}'$ which means u_i is the XOR of two distinct elements in $\{x_0', \ldots, x_{q+1}'\}$. In either case, u_i must lie in $L_i^{\mathbf{x},D}$. Therefore, by Theorem 5.23 in [16], for every $\mathbf{x}' \in \mathcal{X}^{q+2}$ we have

$$[\![\neg\mathsf{TCHN}_{\mathbf{x}'}^{q+1} \xrightarrow{q+2} \mathsf{TCHN}_{\mathbf{x}'}^{q+1}]\!] \leq e(q+2)\sqrt{\frac{5(q+2)(q+3)}{|\mathcal{Y}|}}.$$

Thus, we can bound $\max_{\mathbf{x}'} \|\mathsf{TCHN}_{\mathbf{x}'}^{q+1}\mathsf{cO}_{x_{q+1}'}\neg\mathsf{TCHN}_{\mathbf{x}'}^{q+1}\|$ by the above quantity.

Putting things together, we have

$$p_{\mathcal{A}} \leq \left([\![\bot \xRightarrow{q,k} \mathsf{TCHN}^{q+1}]\!] + e(q+2)\sqrt{\frac{5(q+2)(q+3)}{|\mathcal{Y}|}} + \sqrt{\frac{q+2}{|\mathcal{Y}|}}\right)^2$$

Bounding the first term by Lemma 31, this concludes the proof. ◀

Now, we are ready to prove the main theorem.

Proof of Theorem 24. We finish the proof by reduction. Define the algorithm $\mathcal{D}$ as follows:
1. Run $\mathcal{C}$ and obtain x_0, x_q and x_{q+1}.
2. For $i \in [q-1]$:
   - Make a classical 2-parallel query (x_{i-1}, x_{q+i}) to the random oracle and then obtain $(H(x_{i-1}), H(x_{q+i}))$.
   - Set $x_i := H(x_{i-1}) \oplus x_{i-2}$ and $x_{q+i+1} := H(x_{q+i}) \oplus x_{q+i-1}$.
3. Make a classical 2-parallel query (x_{q-1}, x_{2q}) to the random oracle and then obtain $(H(x_{q-1}), H(x_{2q}))$.
4. Output $x_0, x_1, \ldots, x_{2q+1}$ and $H(x_0), H(x_1), \ldots, H(x_{q-1}), H'(x_q), H(x_{q+1}), \ldots, H(x_{2q})$, where $H'(x_q) := x_{q-1} \oplus x_{q+1}$[16].

---

[15] In the rest of the proof, we switch the variable $\mathbf{x}$ of $\mathsf{TCHN}_{\mathbf{x}}^{q+1}$ into $\mathbf{x}'$ for convenience.

[16] Note that in Step 2 and 3, $\mathcal{D}$ makes a total of $2q$ queries including $x_0, \ldots, x_{2q}$ except x_q.

First, it is trivial that $p_\mathcal{C}$ is equivalent to the probability $p_\mathcal{D}$ that $H'(x_q) = H(x_q)$ and $\mathcal{D}$ outputs a $(2q + 1)$-chain. Now, we calculate the total number of queries made by $\mathcal{D}$. In Step 1, $\mathcal{D}$ makes q k-parallel queries to execute $\mathcal{C}$. In Steps 2 and 3, $\mathcal{D}$ makes q 2-parallel queries. To sum up, $\mathcal{D}$ makes a total of $2q$ k-parallel queries. By Lemma 32, the probability $p_\mathcal{D}$ is at most $F(k, 2q)$. Therefore, this finishes the proof. ◄

Considering the situation in which the algorithm is assigned to a particular starting point $x_0 \in \mathcal{X}$ of the chain, we have the following corollary which is trivially implied by Theorem 24.

▶ **Corollary 33.** *For any k-parallel q-query oracle algorithm $\mathcal{E}$, the probability $p_\mathcal{E}$ (parameterized by k and q) that the algorithm takes a uniformly random $x_0 \in \mathcal{X}$ as input, and outputs $x_q, x_{q+1} \in \mathcal{X}$ satisfying*
▬ *there exist $x_1, \ldots, x_{q-1} \in \mathcal{X}$ such that $H(x_{i-1}) = x_i \oplus x_{i-2}$ for $i \in [q+1]$, where $x_{-1} := 0^n$*
is at most $F(k, 2q) = O(k^4 q^4 / |\mathcal{Y}|)$, where the function F is defined in Lemma 32.

Proof. We finish the proof by reduction. Let $\mathcal{C}$ be the algorithm that first samples $x_0 \in \mathcal{X}$ uniformly at random and invokes $\mathcal{E}(x)$. $\mathcal{C}$ responds to every $\mathcal{E}$'s oracle query by its oracle access directly. Then $\mathcal{C}$ outputs whatever $\mathcal{E}$ outputs.

Let $p_\mathcal{C}$ be the probability defined as in Theorem 24. Since the success of $\mathcal{E}$ implies the success of $\mathcal{C}$, we have $p_\mathcal{C} \geq p_\mathcal{E}$. By Theorem 24 and the construction of $\mathcal{C}$, we have $F(k, 2q) \geq p_\mathcal{C}$, which concludes the proof. ◄

▶ **Remark 34.** Here, we explain the challenging issue of our case. Given only x_0, x_q and x_{q+1}, in order to output the whole chain, the algorithm cannot make the query in parallel but is required to make *adaptive* queries. For example, to reveal the next point $x_1 = H(x_0)$, the algorithm must first query x_0. Therefore, we cannot use Theorem 5.9 in [16] in a black-box way.

7 Quantum Walk on a Line

Our proof of Hamiltonian simulation lower bound relies on the continuous-time quantum walk on a line [14]. We introduce quantum walks on a line in this section.

Consider a particle moving on a graph, which is a line with L vertices. Each vertex on the line is labeled by an integer $1, 2, \ldots L$. We use a quantum state $|j\rangle$ to denote the particle locating at the vertex j. Figure 4(a) illustrates our system, a finite segment with length L. We let the Hamiltonian H_L of the system be the adjacency matrix of the graph. In physics terminology, H_L couples adjacent vertices with the coupling constant 1. We have
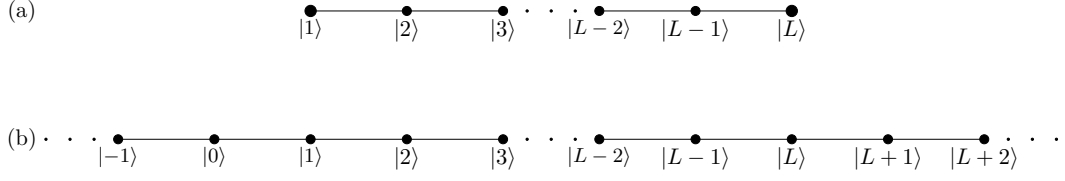
$$H_L = \sum_{j=1}^{L-1} |j\rangle\langle j+1| + |j+1\rangle\langle j|, \tag{29}$$

or

$$H_L = \begin{pmatrix} 0 & 1 & 0 & & & 0 \\ 1 & 0 & 1 & \cdots & \cdots & 0 \\ 0 & 1 & 0 & & & 0 \\ & \vdots & & \ddots & & \vdots \\ & \vdots & & & 0 & 1 \\ 0 & 0 & 0 & \cdots & 1 & 0 \end{pmatrix}.$$

in the $\{|j\rangle\}_{j=1}^{L}$ basis.

(a)

(b)

Figure 4 Quantum walk on a line. (a) Quantum walk on a finite segment with length L. (b) Quantum walks on an infinite line.

The dynamics of the particle are determined by the time evolution operator $e^{-iH_L t}$, where t is the evolution time. We call the dynamics of the system "quantum walk on a line."

We are interested in the dynamics of a particle initially at the end of the line. In other words, we consider the evolution of a particle under H_L with the initial state $|1\rangle$. We have the following result.

▶ **Lemma 35.** *Given a system that evolves under the Hamiltonian H_L described in* (29) *with initial state $|1\rangle$, if the system is measured at time $t \in [0, L/2]$ in the $\{|j\rangle\}_{j=0}^L$ basis with outcome l, the probability that $l > t$ is at least $1/3$.*

Before the formal proof of Lemma 35, we first discuss the general behavior of the quantum walk. Let the particle initially locate at $|k\rangle$ and evolve under the Hamiltonian H_L. When measuring the system at time t in the $\{|j\rangle\}_{j=1}^L$ basis, the probability $P(k, l, t)$ of measurement outcomes being l is

$$P(k, l, t) = \left| \langle l | e^{-iH_L t} | k \rangle \right|^2 . \tag{30}$$

By diagonalizing H_L, we can calculate $P(k, l, t)$ as follows:

$$P(k, l, t) = \left| \langle l | e^{-iH_L t} | k \rangle \right|^2 = \sum_{p,q=1}^L e^{-i(\lambda_p - \lambda_q)t} v_l^{(p)} v_k^{(p)*} v_l^{(q)*} v_k^{(q)}, \tag{31}$$

where λ_p's are the eigenvalues of H_L – each with the corresponding eigenstate $|v^{(p)}\rangle = \sum_{j=1}^L v_j^{(p)} |j\rangle$. The eigenvalues and the eigenstates of H_L have a closed-form expression. That is, $\lambda_p = 2\cos(\frac{p\pi}{L+1})$ and $v_j^{(p)} = \sqrt{\frac{2}{L+1}} \sin(\frac{jp\pi}{L+1})$ [30] [17].

We use the propagation of the wave function of a free particle to analogize the quantum walk.[18] For example, we plot the result of the quantum walk on a segment of length $L = 100$ in Figure 5. The initial state is $|k = 1\rangle$ and we focus on the time interval $t \in [0, L/2]$. Figure 5(a) shows $P(1, l, t)$, the probability of obtaining the measurement outcome $|l\rangle$, for

---

[17] In fact, there is a simpler form of $P(k, l, t)$: when $|l - k|$ is even,

$$P(k, l, t) = \left(\sum_p \cos\left(2t \cos\left(\frac{p\pi}{L+1} \right) \right) \sin\left(\frac{kp\pi}{L+1} \right) \sin\left(\frac{lp\pi}{L+1} \right) \right)^2,$$

and when $|l - k|$ is odd,

$$P(k, l, t) = \left(\sum_p \sin\left(2t \cos\left(\frac{p\pi}{L+1} \right) \right) \sin\left(\frac{kp\pi}{L+1} \right) \sin\left(\frac{lp\pi}{L+1} \right) \right)^2.$$

[18] Consider an extreme case that the distance between two adjacent vertices in space goes to zero and the length L goes to infinity. The system is reduced to free space.

every l at different time t. We see that at time t, the wavefront reaches $l \approx 2t$. Figure 5(b) shows the probability of getting the measurement outcome $|l\rangle$ for a *fixed* l versus time. We see that the probability is extremely small when $t \ll l/2$ and reaches the maximum at $t \approx l/2$. Finally, it behaves like a damped oscillation when $t \gtrsim l/2$. These observations suggest that the wavefront propagates at a constant speed, which gives a hint that the particle reaches the vertex $l = \Theta(t)$ at time t.[19]
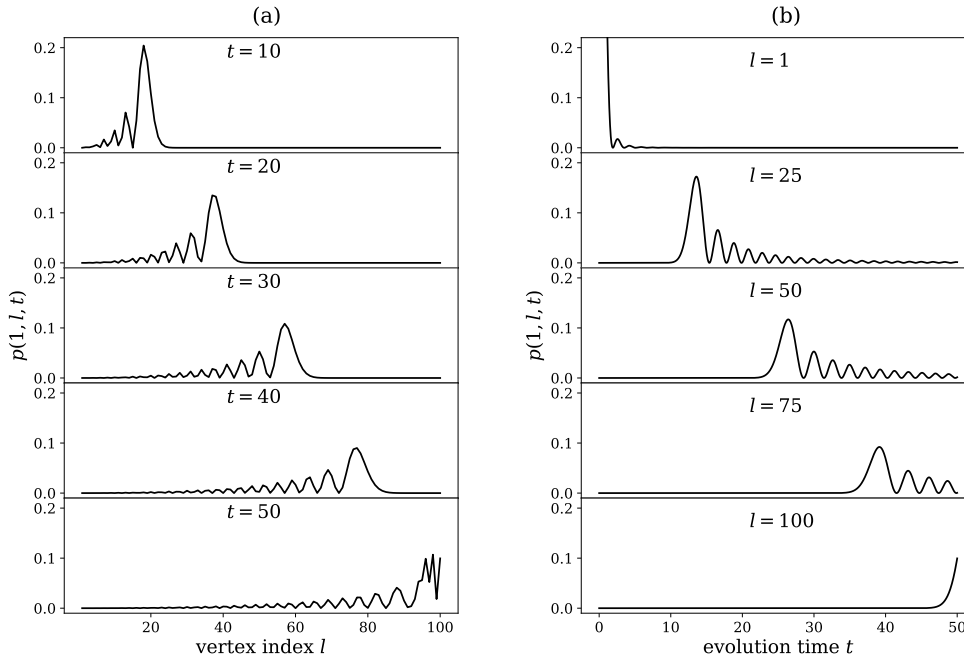
Figure 5 The result of quantum walk on a segment with $L = 100$ for the evolution time $t \in [0, L/2]$. The initial state is $|1\rangle$. (a) The probability of getting the outcomes $|l\rangle$ for every node l at the evolution time $t = 10, 20, 30, 40$ and 50 respectively. (b) The probability of getting the outcomes $l = 1, 25, 50, 75$ and 100 versus evolution time $t \in [0, L/2]$.

Next, we are going to prove Lemma 35. We take another approach instead of diagonalizing H_L directly. We follow the approach in [14]. Similar to solving "the particle in a box model" in quantum mechanics, we first find the homogeneous solution in free space and then find the particular solution that satisfies the boundary conditions and the initial conditions. (See, for example, [34].)

Consider the quantum walk on an infinite line which is illustrated in Figure 4(b). The Hamiltonian of the quantum walk on an infinite line is defined by

$$H_\infty := \sum_{j=-\infty}^{\infty} |j+1\rangle\langle j| + |j\rangle\langle j+1|, \tag{32}$$

---

[19] This corresponds to the fact that the uncertainty of the position of a free particle is linear in t. See, for example, [34].

and we define the propagator

$$G(k, l, t) := \langle l | e^{-iH_\infty t} | k \rangle. \tag{33}$$

The (sub-normalized) eigenstate of H_∞ is the momentum state $|p\rangle$. The momentum state has the following property

$$\langle j | p \rangle = e^{ipj}, -\pi \le p \le \pi. \tag{34}$$

The corresponding eigenvalue of $|p\rangle$ is $E_p = 2 \cos p$. Hence, we have

$$\langle l | e^{-iH_\infty t} | k \rangle = \int_{-\pi}^{\pi} dp e^{-i2t \cos p + ip(l-k)} = i^{(l-k)} J_{l-k}(2t), \tag{35}$$

where $J_n(\cdot)$ is the Bessel function of order n. (See (1).)

Now we are ready to calculate the propagator of the quantum walk on a *finite* segment. We use $\widetilde{G}(k, l, t) := \langle l | e^{-iH_L t} | k \rangle$ to denote the propagator of the quantum walk on a finite segment. The propagator $\widetilde{G}$ is a superposition of G and $\widetilde{G}$ that satisfies the boundary conditions: $\widetilde{G}(k, 0, t) = 0 = \widetilde{G}(k, L + 1, t)$, and the initial condition $\widetilde{G}(k, l, 0) = \delta_{kl}$.

The solution is

$$\widetilde{G}(1, l, t) = \sum_{m=-\infty}^{\infty} G(1, l + 2m(L+1), t) - G(1, -l + 2m(L+1), t). \tag{36}$$

The above equation (36) can be interpreted as the wave reflecting between the boundaries $j = 0$ and $j = L + 1$.

We set the starting point $j = 1$. In the time interval that we are interested in, namely, $t \in [0, L/2]$, we have $G(1, \pm l + 2m(L+1), t) = J_{2m(L+1)\pm l - 1}(2t)$ is exponentially small in L for $m \ne 0$. This is because the order $|2m(L+1) \pm l - 1| > L$ for $m \ne 0$ and the argument $2t \le L$ for $t \le L/2$. (See (4).)

Thus,

$$\begin{aligned}
\widetilde{G}(1, l, t) &\approx G(1, l, t) - G(1, -l, t) \\
&= i^{l-1} J_{l-1}(2t) - i^{-(l+1)} J_{-(l+1)}(2t) \\
&= i^{l-1} J_{l-1}(2t) - (i^{-(l+1)})(-1^{l+1}) J_{l+1}(2t) \\
&= i^{l-1} J_{l-1}(2t) - (-i)^{l+1}) J_{l+1}(2t) \\
&= i^{l-1} (J_{l-1}(2t) - (-i)^2 J_{l+1}(2t)) \\
&= i^{l-1} (J_{l-1}(2t) + J_{l+1}(2t)) \\
&= i^{l-1} \frac{l}{t} J_l(2t).
\end{aligned} \tag{37}$$

The third equation is due to the relation of negative order (2) of the Bessel function, and the last equation uses the recursion property (3) of the Bessel function. Then we have

$$P(1, l, t) = \left| \widetilde{G}(1, l, t) \right|^2 \approx \left(\frac{l}{t} \right)^2 J_l^2(2t). \tag{38}$$

As a remark, the probability $P(1, l, t)$ is almost independent of L when $t \in [L/2]$. It can be interpreted as the following: before the wavefront reaches the boundary, the wave propagates as in free space. Finally, we prove Lemma 35.

Proof of Lemma 35. We directly calculate the probability $\sum_{l=1}^{\lfloor t \rfloor} P(1, l, t)$ as follows:

$$\sum_{l=1}^{\lfloor t \rfloor} P(1, l, t) = \sum_{l=1}^{\lfloor t \rfloor} \left(\frac{l}{t}\right)^2 J_l^2(2t) \leq \sum_{l=1}^{\lfloor t \rfloor} \left(\frac{l}{t}\right)^2 \frac{2}{\pi} \frac{1}{l} = \frac{2}{\pi} \sum_{j=l}^{\lfloor t \rfloor} \frac{l}{t^2} \leq \frac{2}{\pi},$$

where the first equation follows from (38) and the second inequality follows from Lemma 15. As a result, we conclude that

$$\sum_{l=\lceil t \rceil}^{L} P(1, l, t) = 1 - \sum_{l=1}^{\lfloor t \rfloor} P(1, l, t) > \frac{1}{3}. \qquad \blacktriangleleft$$

8   No Fast-forwarding in Oracle Model: Unconditional Result

In this section, we are going to investigate the parallel lower bound of Hamiltonian simulation in the oracle model. In the oracle model, the Hamiltonian is expressed by a Hermitian matrix. There are many algorithms that can efficiently simulate a Hamiltonian in the oracle model if the Hamiltonian matrix is sparse [6–8, 13, 27, 28]. As a result, we are interested in the lower bound of simulating a sparse Hamiltonian. Besides, we normalize the Hamiltonian by setting the absolute value of every element of the Hamiltonian to be at most 1. The sparse Hamiltonian is defined as follows.

▶ **Definition 36** (Sparse Hamiltonian). *Let $H \in \mathbb{C}^{N \times N}$ denote a Hamiltonian acting on the Hilbert space with dimension N. We say H is d-sparse if there are at most d nonzero entries in every row.*

In the oracle setting, the simulation algorithm can only obtain the description of the Hamiltonian via oracle queries. In most of the models of the algorithms, there are two oracles that can be accessed: First, the *entry oracle*, denoted by $\mathcal{O}_H$, answers the value of the matrix element. Second, the *sparse structure oracle*, denoted by $\mathcal{O}_L$, answers the index of the nonzero entry. Let the Hamiltonian H that we want to simulate be acting on an N-dimensional Hilbert space and be d-sparse. When the entry oracle $\mathcal{O}_H$ is queried on the index (j, k) where $j, k \in [N]$, it returns the element value H_{jk}. When the sparse structure oracle is queried on (j, s) where $j \in [N]$ and $s \in [d]$, it returns k where H_{jk} is the s-th nonzero entry of the j-th row.

The algorithm can query these two oracles in superposition respectively. In the standard quantum oracle model, these two oracles are written as:

$$\mathcal{O}_H |j, k, z\rangle = |j, k, z \oplus H_{jk}\rangle, \tag{39}$$

and

$$\mathcal{O}_L |j, s\rangle = |j, k\rangle, \tag{40}$$

where k is the index of the s-th nonzero entry in the j-th row.

We are going to prove that simulating a quantum system for evolution time t requires at least $\Omega(t)$ parallel quantum queries. We have the following result.

▶ **Theorem 37** (Simulation lower bound in the oracle model). *For any integer n, any polynomial $T(\cdot)$ and $p = \mathrm{poly}(n)$, there exists a time-independent Hamiltonian $H \in \mathbb{C}^{(2^n T(n)) \times (2^n T(n))}$ satisfies the following. For any quantum algorithm that can make p-parallel queries to the entry oracle $\mathcal{O}_H$ (defined in (39)) and the sparse structure oracle $\mathcal{O}_L$ (defined in (40)), simulating H for an evolution time $t \in [0, T(n)/2]$ within an error $\epsilon \leq 1/4$ needs at least $\Omega(t)$ p-parallel queries to $\mathcal{O}_H$ and $\mathcal{O}_L$ in total. Furthermore, H is 2-sparse and $|H_{jk}| \leq 1$ for every $j, k \in [2^n T(n)]$.*

Theorem 37 can be interpreted as simulating a system with $n + O(\log n)$ qubits for an evolution time $t < \text{poly}(n)$ cannot be fast-forwarded.

Before the formal proof of Theorem 37, we first sketch our proof strategy. We modify the proof of the query lower bound in [6]. In [6], the parity problem is reduced to the Hamiltonian simulation problem. In particular, it is shown that if one can fast-forward the Hamiltonian simulation, then one can find the parity of an N-bit string with $o(N)$ queries. However, this technique cannot be extended to prove the parallel lower bound since finding the parity of a string is not parallel-hard. Instead, we reduce the permutation chain problem, of which the parallel hardness was already proven in Section 5, to the Hamiltonian simulation. We are going to show that there exists a specific Hamiltonian such that simulating the Hamiltonian implies solving the permutation chain problem.

We restate the permutation chain problem and its hardness below.

▶ **Definition 38** (Permutation chain). *Let $n \in \mathbb{N}$ and $p, L = \text{poly}(n)$. For each $j \in [L]$, let $\Pi_j : \{0,1\}^n \to \{0,1\}^n$ be a random permutation and let Π_j^{-1} be the inverse of Π_j. Let $f^{(j)}(\cdot)$ denote $\Pi_j(\Pi_{j-1}(\cdots \Pi_1(\cdot)))$. A quantum algorithm can make p-parallel query to both Π_j and Π_j^{-1} for each $j \in [L]$ respectively and is asked to output $x_q \in \{0,1\}^n$ such that $x_q = f^{(q)}(0^n)$, where $q \in [L]$.*

▶ **Corollary 39** (Hardness of permutation chain). *Let $n \in \mathbb{N}$, and $p, L = \text{poly}(n)$. For each $j \in [L]$, let Π_j and Π_j^{-1} be a random permutation over n-bit strings and its inverse. Let $f^{(j)}(\cdot) := \Pi_j(\Pi_{j-1}(\cdots \Pi_1(\cdot)))$ be the function defined in Definition 38. For any $t, q \in [L]$ and any quantum algorithm $\mathcal{A}$ that makes t p-parallel queries to Π_j and Π_j^{-1}, the probability that $\mathcal{A}$ outputs $x_q \in \{0,1\}^n$ satisfying $x_q = f^{(q)}(0^n)$ and $t < q$ is negligible in n.*

Proof. Let $\bar{x}_q := f^{(q)}(0^n)$ for each $q \in [L]$. The probability that $\mathcal{A}$ outputs $\bar{x}_q$ such that $t < q$ is given by

$$\sum_{j=t+1}^{L} \Pr[\mathcal{A} \text{ outputs } \bar{x}_j].$$

By Theorem 18, for any quantum algorithm A that makes t p-parallel queries to Π_j and Π_j^{-1}, the probability that $\mathcal{A}$ outputs x_j such that $j > t$ is $O(t\sqrt{p/2^n})$. Hence, the probability

$$\sum_{j=t+1}^{L} \Pr[\mathcal{A} \text{ outputs } \bar{x}_j] = \text{poly}(n) \cdot O\left(t\sqrt{\frac{p}{2^n}}\right)$$

is negligible in n. ◀

Similar to [6], we use quantum walk on a graph to solve the underlying hard problem. We construct a graph that consists of L columns where there are 2^n vertices in each column. Each vertex in the j-th column is labelled by (j, x), where $j \in \{0, 1, \ldots L\}$ and $x \in \{0,1\}^n$. The label is translated as follows: after j queries, the output string is x. The vertices in the j-th column are only adjacent to the vertices that are in the $(j \pm 1)$-th columns. Furthermore, the vertices (j, x) and $(j + 1, x')$ (resp., $(j - 1, x')$) are adjacent if and only if $x' = \Pi_{j+1}(x)$ (resp. $\Pi_j^{-1}(x)$). Because each Π_j is a permutation, the graph consists of 2^n disconnected lines of length L. If the vertices (j, x) and $(0, x_0)$ are connected, it holds that $x = f^{(j)}(x_0)$.

In Figure 6, we presents a toy example: let $\Pi_j : \{0,1\}^2 \to \{0,1\}^2$ for each $j \in [L]$ and each $\Pi_j = \Pi$ has the same truth table, i.e.,

$\Pi(00) = (01), \ \Pi(01) = (10), \ \Pi(10) = (11), \ \text{and} \ \Pi(11) = (00).$
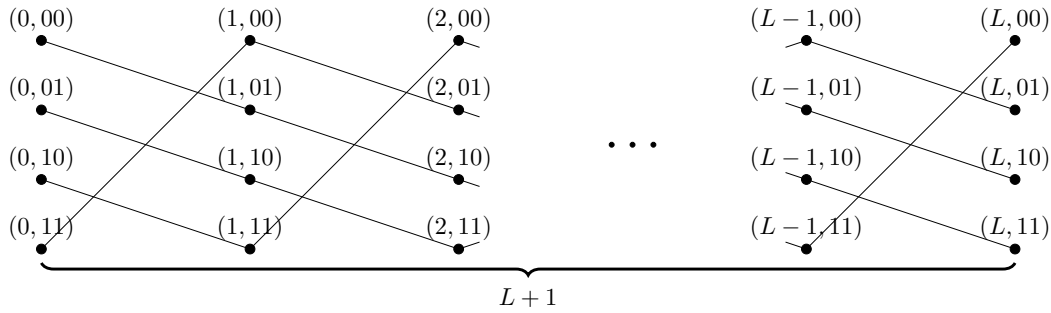
Figure 6 Using quantum walk to solve the permutation chain problem: a toy example.

We let the Hamiltonian H that determines the behavior of the quantum walk be the adjacency matrix of the graph.[20] That is,

$$H = \sum_{j=0}^{L-1} \sum_{x \in \{0,1\}^n} |j+1, \Pi_j(x)\rangle\langle j, x| + |j, x\rangle\langle j+1, \Pi_j(x)|$$

$$= \sum_{x_0 \in \{0,1\}^n} \sum_{j=0}^{L-1} |j+1, f^{(j+1)}(x_0)\rangle\langle j, f^{(j)}(x_0)| + |j, f^{(j)}(x_0)\rangle\langle j+1, f^{(j+1)}(x_0)|. \tag{41}$$

Because two vertices on different lines are decoupled, we have the following observation.

▶ **Observation 40.** *If the random walk starts at the vertex $(0, x_0)$, then it always walks on the same line. To be more precise, if a system evolves under the Hamiltonian H described in (41) and the initial state is $|0, x_0\rangle$, then at any time t, the quantum state of the system is in the subspace* $\text{Span}\left(\left\{|j, f^{(j)}(x_0)\rangle\right\}_{j=0}^{L}\right)$.

Observation 40 can be verified by taking the Taylor expansion of the time evolution operator: $e^{-iHt} = \sum_{k=0}^{\infty} (-iHt)^k / k!$.

To solve the permutation chain problem, we use a Hamiltonian simulation algorithm to simulate the quantum walk under the Hamiltonian H with initial state $|0, 0^n\rangle$. When we measure the system at time t and get the outcome (q, x). The string x is a potential solution to the permutation chain problem. We aim to prove the following two statements. First, the oracles $\mathcal{O}_H$ and $\mathcal{O}_L$ can be simulated efficiently by Π_j and Π_j^{-1}. Second, the probability of getting a measurement outcome (q, x) at time t such that $q \geq t$ is high. Combining these two statements, we have the following conclusion. If an algorithm can simulate H for an evolution time t with $o(t)$ queries, then we can solve the permutation chain problem with $o(t)$ queries as well. However, this violates the hardness of the permutation chain problem.

Now we are ready to present the formal proof of Theorem 37.

Proof of Theorem 37. We construct a time-independent Hamiltonian H acting on a $2^n(L+1)$-dimensional Hilbert space where $L + 1 = f(n)$. The basis vector of the $2^n(L+1)$-dimensional Hilbert space is denoted by $|j, x\rangle$ where $j \in \{0, 1, \ldots, L\}$ and $x \in \{0, 1\}^n$. The element of H is defined as follows.

$$\langle j', x'|H|j, x\rangle = \begin{cases} 1, & \text{if } j' = j+1 \text{ and } x' = \Pi_{j+1}(x) \\ 1, & \text{if } j' = j-1 \text{ and } x' = \Pi_j^{-1}(x) \\ 0, & \text{otherwise.} \end{cases} \tag{42}$$

---

[20] Our Hamiltonian is different from that appears in [6], in which the graph is weighted.

Notice that the Hamiltonian H is 2-sparse and the absolute value of every matrix element is at most 1.

We are going to show the following. Suppose $\mathcal{A}$ can simulate H for an evolution time $t \in [0, (L+1)/2]$ within an error $\epsilon < 1/4$ by making $o(t)$ p-parallel queries to $\mathcal{O}_H$ and $\mathcal{O}_L$. Then we can construct a reduction $\mathcal{R}$ that makes $o(t)$ p-parallel queries to Π_j and Π_j^{-1} and outputs a pair (x_0, x_q) such that $x_q = f^{(q)}(x_0)$ and $q > t$ with constant probability. The reduction $\mathcal{R}$ is described as follows:

1. Run the Hamiltonian simulation algorithm $\mathcal{A}$ on inputs the Hamiltonian H, the evolution time $t \in [0, L/2]$ and the initial state $|0, 0^n\rangle$.
   - When $\mathcal{A}$ queries $\mathcal{O}_H$ on the index $((j, x), (j', x'))$, the reduction $\mathcal{R}$ returns the response by the following rules.
     - If $j' = j + 1$ and $x' = \Pi_{j+1}(x)$, then $\mathcal{R}$ returns 1.
     - If $j' = j - 1$ and $x' = \Pi_j^{-1}(x)$, then $\mathcal{R}$ returns 1.
     - Otherwise, $\mathcal{R}$ returns 0.
   - When $\mathcal{A}$ queries $\mathcal{O}_L$ on $((j, x), s)$, reduction $\mathcal{R}$ returns the response by the following rules.
     - If $j = 0$ and $s = 1$, then $\mathcal{R}$ returns $(1, \Pi_1(x))$.
     - If $j = L$ and $s = 1$, then $\mathcal{R}$ returns $(L - 1, \Pi_L^{-1}(x))$
     - If $j \neq 0, L$ and $s = 1$, then $\mathcal{R}$ returns $(j - 1, \Pi_j^{-1}(x))$.
     - If $j \neq 0, L$ and $s = 2$, then $\mathcal{R}$ returns $(j + 1, \Pi_{j+1}(x))$.

2. Measure the system in the $\{|j, x\rangle\}$ basis and obtain the outcome (q, x_q).

3. Output x_q.

Note that answering a query to the entry oracle $\mathcal{O}_H$ can be implemented by $O(1)$ queries to Π_j and Π_j^{-1}. Similarly, the sparse structure oracle $\mathcal{O}_L$ can be simulated by $O(1)$ queries to Π_j and Π_j^{-1} as well.

Next, we analyze the evolution under H. Let us define another Hamiltonian $H|_0$ restricted to the subspace $\text{Span}\left(\{|j, f^{(j)}(0)\rangle\}_{j=0}^{L}\right)$:

$$H|_0 = \sum_{j=0}^{L-1} |j + 1, f^{(j+1)}(0)\rangle\langle j, f^{(j)}(0)| + |j, f^{(j)}(0)\rangle\langle j + 1, f^{(j+1)}(0)|.$$

By Observation 40, the time evolution under $H|_0$ is equivalent to the time evolution under H with the initial state $|0, 0^n\rangle$.

We first consider a perfect Hamiltonian simulation algorithm $\widetilde{\mathcal{A}}$ that outputs the state $|\widetilde{\psi}\rangle := e^{-iHt}|0, 0^n\rangle = e^{-iH|_0 t}|0, 0^n\rangle$. In Step 2, the measurement outcome (q, x_q) satisfies $x_q = f^{(q)}(0^n)$. Then by Lemma 35, the probability that the measurement outcome satisfies $q > t$ is at least $1/3$.

Next, we consider the general simulation algorithm that outputs a state $|\psi\rangle$ such that $\Delta(|\psi\rangle\langle\psi|, |\widetilde{\psi}\rangle\langle\widetilde{\psi}|) \leq 1/4$. By the property of the trace distance, the difference in probabilities that $\mathcal{R}$ outputs a correct outcome by measuring $|\psi\rangle$ and $|\widetilde{\psi}\rangle$ is at most $1/4$. As a result, $\mathcal{R}$ outputs the accepted string x_q with probability at least $1/3 - 1/4 = 1/12$.

Combining everything together, if $\mathcal{A}$ simulates H for time t within $\epsilon \leq 1/4$ by making $o(t)$ p-parallel queries, then $\mathcal{R}$ will output $x_q = f^{(q)}(0^n)$ such that $q > t$ with constant probability by making $o(t)$ p-parallel queries. This contradicts Corollary 39. ◀

9   No Fast-forwarding in Plain Model

In this section, we are going to investigate the parallel lower bound of Hamiltonian simulation in the plain model. In the plain model, we are interested in the Hamiltonians that have a succinct description. Typically, we consider the *local Hamiltonians.*

▶ **Definition 41** (Local Hamiltonian). *We say a Hamiltonian H that acts on n qubits is k-local if H can be written as*

$$H = \sum_j H_j,$$

where each H_j acts non-trivially on at most k qubits.

The *geometrically local* Hamiltonians are another kind of Hamiltonians that often appear in physics models. A geometrically local Hamiltonian is a local Hamiltonian with more constraints. For a geometrically local Hamiltonian written by $H = \sum_j H_j$, each term H_j acts non-trivially on the qubits that are near in space. We are especially interested in one-dimensional geometrically local Hamiltonians.

▶ **Definition 42** (One-dimension geometrical local Hamiltonians). *Let a system consist of n qubits that are aligned in space and each qubit is labeled by an integer $l \in [n]$. Let $H = \sum_j H_j$ be a k-local Hamiltonian that acts on n qubits. We say H is an* one-dimension geometrically local *Hamiltonian if each H_j acts non-trivially on at most k consecutive indices.*

For example, consider Hamiltonians H_1 and H_2 acting on four qubits defined as follows:

$$H_1 := \sigma^X \otimes \sigma^Z \otimes I \otimes I + \sigma^Z \otimes I \otimes I \otimes \sigma^Z + I \otimes I \otimes \sigma^X \otimes I,$$

and

$$H_2 := \sigma^Z \otimes \sigma^Z \otimes I \otimes I + I \otimes \sigma^Z \otimes \sigma^Z \otimes I + I \otimes I \otimes \sigma^Z \otimes \sigma^Z,$$

where σ^X and σ^Z are Pauli operators and I is the identity operator. Hamiltonian H_1 is 2-local but not geometrically local, but Hamiltonian H_2 is geometrically local. We normalize the Hamiltonian by setting the spectral norm $\|H_j\| = O(1)$ for each j.

Having a succinct description gives the simulation algorithm more power than in the oracle model. In this sense, we obtain a stronger lower bound. On the other hand, our lower bound in the plain model relies on computational assumptions, which weakens the result. For our lower bound, we need to assume an iterative parallel-hard function, which is slightly modified from the definition of an iterative sequential function by Boneh et al. [9].

▶ **Definition 43** (Iterative parallel-hard functions/puzzles). *A function $f \colon \mathbb{N} \times \hat{X} \to X$ where $\hat{X} \in X$ and $|\hat{X}| = 2^{\theta(\lambda)}$ is a (post-quantum) (s, d)-iterated parallel-hard function if there exists a function $g \colon X \to X$ such that*
- *g can be computed by a quantum circuit with width λ and size $s(\lambda)$. Without loss of generality, we can let $s(\lambda) = \Omega(\lambda)$*
- *$f(k, x) = g^{(k)}(x)$.*
- *For all sufficiently large $k = 2^{o(\lambda)}$, for any quantum circuit C with depth less than $d(k)$ and size less than $\mathrm{poly}(t, d(k), \lambda)$,*

$$\Pr[C(x) = f(k, x) \mid x \leftarrow \hat{X}] \leq \mathrm{negl}(\lambda)$$

Without loss of generality, we assume that d is non-decreasing.

We say that f forms a (post-quantum) (s, d)-iterated parallel-hard puzzle if it only satisfies a weaker version of the third requirement as follows:

- *For all $k = 2^{o(\lambda)}$, for any uniform quantum circuit C with depth less than $d(k)$ and size less than $\mathrm{poly}(t, d(k), \lambda)$,*

$$\Pr[C(x) = f(k', x) \text{ for some } k' \geq k/2 \mid x \leftarrow \hat{X}] \leq \mathrm{negl}(\lambda).$$

Note that an (s, d)-iterated parallel-hard function is directly an (s, d')-iterated parallel-hard puzzle, where $d'(x) := d(x/2)$.

Under the (quantum) random oracle heuristic [4, 10], such parallel-hard puzzles can be heuristically obtained by instantiating the twisted hash chain with a cryptographic hash function.

▶ **Assumption 44.** *With the random oracle heuristic, we can assume that the standard instantiation of the twisted hash chain is parallel-hard by Corollary 33. Assuming the cryptographic hash function h in the instantiation can be implemented by circuits of size $s(\lambda)$ on λ-bit inputs, the twisted hash chain directly gives an $(s + O(\lambda), d)$ iterative parallel-hard function with $d(x) := x - 1$, which is an $(s + O(\lambda), d)$ iterative parallel-hard puzzle with $d(x) := \lfloor \frac{x}{2} \rfloor - 1$*

We present the simulation lower bound for the local Hamiltonians in the following theorem.

▶ **Theorem 45** (Simulation lower bound for local Hamiltonians in the plain model). *Assuming an (s, d)-iterated parallel-hard puzzle, for any integer n, there exists a time-independent c-local Hamiltonian H acting on $n + (2s(n)T(n))^{1/c}$ qubits such simulating H for an evolution time $t \in [0, s(n)T(n)]$ with error $\epsilon < 1/4$ needs a $(d(\lfloor t/2s(n) \rfloor) - O(s(n)))$-depth circuit, where $T(\cdot)$ is an arbitrary polynomial and c is a constant.*

We also have the lower bound for simulating geometrically local time-dependent Hamiltonians.

▶ **Theorem 46** (Simulation lower bound for geometrically local Hamiltonians in the plain model). *Assuming an (s, d)-iterated parallel-hard function, for any integer n, there exists a piecewise-time-independent 1-D geometrically 2-local Hamiltonian H acting on n qubits such that simulating H for an evolution time $t \in [0, ns(n)T(n)]$ with error $\epsilon(n) \leq 1 - 1/\mathrm{poly}(n)$ needs a $\left(d(\lfloor \frac{t}{ns(n)} \rfloor) - O(ns(n)) - \mathrm{poly}(\log(n), \log\log(1/\epsilon'(n))) \right)$-depth circuit, where $T(\cdot)$ is an arbitrary polynomial and $\epsilon'(n) < 1 - \epsilon(n) - 1/\mathrm{poly}(n)$.*

We sketch our proof strategy as follows. The main idea is, again, to reduce the hard problem to the Hamiltonian simulation problem. First, we consider a quantum circuit C that computes an (s, d)-iterated parallel-hard puzzle, which according to the definition, can be written as a sequential composition of λ-qubit $s(\lambda)$-sized circuits. Then we construct a Hamiltonian H_{circuit} to implement the circuit C by the circuit to Hamiltonian reduction technique. The circuit to time-independent reduction is introduced in Section 9.1, and the circuit to time-dependent reduction is introduced in Section 9.3. Finally, we use the Hamiltonian simulation algorithm to simulate the Hamiltonian H_{circuit}. If we can fast-forward the Hamiltonian evolution under H_{circuit}, then we can break the depth guarantee provided by the iterated parallel-hard puzzle.

9.1   Circuit to time-independent Hamiltonian

Feynman suggested that we can implement a quantum circuit (which was called reversible computation at his time) by a time-independent Hamiltonian [21]. About a decade later, Childs and Nagaj provided rigorous analyses for the implementation [15, 30]. The idea is to introduce an extra register, which is called the clock register $\mathcal{H}_{\text{clock}}$, associated with the circuit register $\mathcal{H}_{\text{circuit}}$ to record the progress of the quantum circuit. After introducing the clock register, we define a state $|\psi_j\rangle = |\phi_j\rangle_{\text{circuit}} \otimes |\gamma_j\rangle_{\text{clock}}$ to indicate that the j-th steps outcomes is $|\phi_j\rangle$. We can construct a Hamiltonian acting on $\mathcal{H}_{\text{circuit}} \otimes \mathcal{H}_{\text{clock}}$ such that during the evolution, the system is in $\text{Span}(\{|\psi_j\rangle\})$. When we measure the clock register $\mathcal{H}_{\text{clock}}$ and get the outcome γ_k, the quantum state in the circuit register $\mathcal{H}_{\text{circuit}}$ collapses to $|\phi_k\rangle$. And then we obtain the k-th step outcome of the quantum circuit. Similar techniques appear in the proof of QMA completeness [25] and universality of adiabatic computation [1].

In [30], Nagaj proved that for any quantum circuit C with L quantum gates, there is a Hamiltonian H_{circuit} such that evolving the system under the Hamiltonian H_{circuit} for time $O(L)$ and then measuring the system, we can get the final state of the quantum circuit C with high probability. We extend Nagaj's result. In this section, we are going to prove that if the system evolves under H_{circuit} for time $t \in [0, L/2]$, we can get $|\psi_j\rangle$ where $j \geq t$ with high probability. Our method is slightly different from Nagai's. In [30], the evolution time is uniformly sampled, while we have an explicit evolution time. Another difference is that in [30] it needs to pad $O(L)$ dummy identity gates at the end of the quantum circuit to amplify the probability of getting the output state. In our construction, padding is not required.

Let a quantum circuit C which acts on the register $\mathcal{H}_{\text{circuit}}$ consist of a sequence of g quantum gates $U_1, U_2, \ldots, U_L$. Namely,

$$C = U_L U_{L-1} \cdots U_1.$$

After introducing the clock register $\mathcal{H}_{\text{clock}}$, and the clock state, which is a family of orthonormal states $\{|\gamma_j\rangle\}_{j=0}^{L}$ where each $|\gamma_j\rangle \in \mathcal{H}_{\text{clock}}$, we construct the following Hamiltonian

$$H_{\text{circuit}} := \sum_{j=1}^{L} H_j, \tag{43}$$

where

$$H_j := U_j \otimes |j\rangle\langle j-1| + U_j^\dagger \otimes |j-1\rangle\langle j|. \tag{44}$$

Let the input state of the circuit be $|\phi_0^{(0)}\rangle$, and let $|\phi_j^{(0)}\rangle$ denote the quantum state of j-th step. That is, $|\phi_j^{(0)}\rangle = U_j U_{j-1} \cdots U_1 |\phi_0^{(0)}\rangle$. Define the state

$$|\psi_j^{(0)}\rangle := |\phi_j^{(0)}\rangle \otimes |\gamma_j\rangle, \tag{45}$$

which is the state after j steps of C.

Let $\{|\phi_0^{(1)}\rangle, \ldots, |\phi_0^{(N-1)}\rangle\}$ be the states that are orthogonal to $|\phi_0^{(0)}\rangle$ where N denotes the dimension of $\mathcal{H}_{circuit}$. Besides, let $|\phi_j^{(m)}\rangle := U_j U_{j-1} \cdots U_1 |\phi_0^{(m)}\rangle$ and $|\psi_j^{(m)}\rangle := |\phi_j^{(m)}\rangle \otimes |\gamma_j\rangle$ where $m \in [N-1]$. We have

$$
H_{\text{circuit}}|\psi_j^{(m)}\rangle = \begin{cases} |\psi_{j+1}^{(m)}\rangle & , \text{ if } j = 0, \\ |\psi_{j-1}^{(m)}\rangle + |\psi_{j+1}^m\rangle & , \text{ if } j = 1, \ldots, L-1, \\ |\psi_{j-1}^{(m)}\rangle & , \text{ if } j = L, \end{cases}
$$

for any $m \in \{0, 1, \ldots, N-1\}$.

Again, the evolution under H_{circuit} can be viewed as quantum walks on a graph. We construct a graph illustrated in Figure 7 whose adjacency matrix is H_{circuit}.
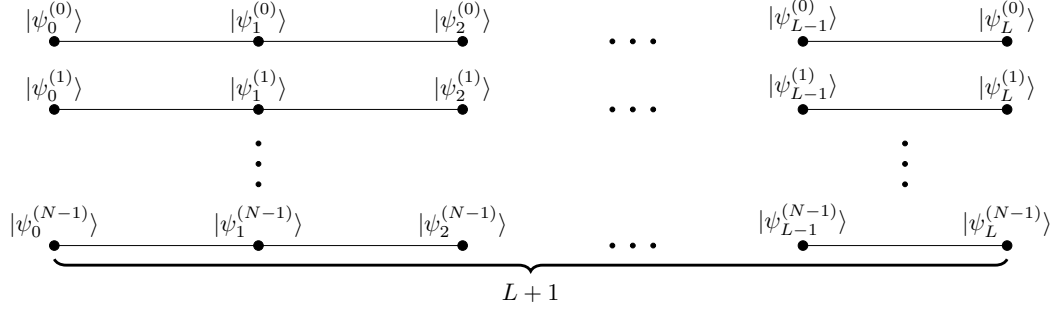
Figure 7 The quantum walk on a graph for the circuit to Hamiltonian reduction.

We have the following lemma.

▶ **Lemma 47.** *A system evolves under the Hamiltonian H_{circuit} described in (43) with the initial state $|\psi_0^{(0)}\rangle$ described in (45). If the clock register is measured at time $t \in [0, L/2]$ in the $\{\gamma_j\}$ basis and get the outcome l, the probability that the circuit register collapses to $|\phi_l^{(m)}\rangle$ where $m = 0$ and $l > t$ is at least $1/3$.*

Proof. By the similar argument of Observation 40, we define another Hamiltonian $H|_{\psi^{(0)}}$ restricted to the subspace $\text{Span}\left(\{|\psi_j^{(0)}\rangle\}_{j=0}^L\right)$:

$$
H|_{\psi^{(0)}} := \sum_{j=0}^{L-1} |\psi_{j+1}^{(0)}\rangle\langle\psi_j^{(0)}| + |\psi_j^{(0)}\rangle\langle\psi_{j+1}^{(0)}|. \tag{46}
$$

If the system is initially at $|\psi_0^{(0)}\rangle$, the time evolution under H_{circuit} is the same as the time evolution under $H|_{\psi^{(0)}}$. As a result, we have $m = 0$ for any time t.

Because $|\psi_j^{(0)}\rangle = |\phi_j^{(0)}\rangle \otimes |\gamma_j\rangle$, the measurement results of measuring clock register in $\{|\gamma_j\rangle\}$ basis is the same as measuring the entire system in $\{|\psi_j^{(0)}\rangle\}$ basis. The probability that $l > t$ can be obtained directly from Lemma 35. This finishes the proof. ◀

Next, we present our construction of the clock state.

▶ **Lemma 48.** *For all $c, T \in \mathbb{N}$, there exists a construction that implements the clock state for time T with locality c and at most $O(T^{1/(c-1)})$ qubits.*

Proof. Let n be the smallest integer such that $\binom{n}{c-1} \geq T$. Thus, $n = O(T^{1/(c-1)})$. Consider the system that consists of n qubits indexed from 1 to n. Consider the Johnson graph $J_{n,c-1}$ (see Definition 16), for each node $S \subseteq [n]$, define the n-qubit state $|S\rangle$ as

$$|S\rangle := \bigotimes_{i=1}^{n} |I_S(i)\rangle_i,$$

where the subscript i denotes the register of the i-th qubit; $I_S : [n] \to \{0,1\}$ is the indicator function that equals 1 if $i \in S$ and 0 otherwise.

Choose an arbitrary Hamiltonian path of $J_{n,c-1}$, denoted by $(S_1, S_2, \ldots, S_{\binom{n}{c-1}})$. Now, each time $j \in [\binom{n}{c-1}]$ corresponds to the n-qubit state $|S_j\rangle$.

For every $j \in \left[\binom{n}{c-1} - 1\right]$, the time transition $|j+1\rangle\langle j|$ is implemented by

$$E_{j \to j+1} := \bigotimes_{i=1}^{n} P_i,$$

where

$$P_i := \begin{cases} |1\rangle\langle 1|_i, & \text{if } i \in S_j \cap S_{j+1} \\ |1\rangle\langle 0|_i, & \text{if } i \in S_{j+1} \setminus S_j \\ |0\rangle\langle 1|_i, & \text{if } i \in S_j \setminus S_{j+1} \\ I, & \text{otherwise.} \end{cases}$$

It is easy to see that the transitions are c-local due to the definition of $J_{n,c-1}$. It remains to check the correctness of the transitions. That is, for every $j, j' \in \left[\binom{n}{c-1} - 1\right]$, they should satisfy

$$E_{j \to j+1}|S_{j'}\rangle = \begin{cases} |S_{j'+1}\rangle, & \text{if } j = j' \\ 0, & \text{otherwise.} \end{cases}$$

When $j = j'$, the equality holds since there is an edge between S_j and S_{j+1}. When $j \neq j'$, there must exist an $i^* \in S_j$ such that $i^* \notin S_{j'}$. Therefore, $P_{i^*} = |0\rangle\langle 1|_{i^*}$ will vanish $|S_{j'}\rangle$ because the i^*-th qubit of $|S_{j'}\rangle$ is in the state $|0\rangle_i$. This verifies the correctness.   ◄

9.2   Proof of the lower bound for local Hamiltonians

Proof of Theorem 45. From an (s,d)-iterated parallel-hard puzzle $f(k,x) = g^{(k)}(x)$, we here show how to construct a time-independent $(c+2)$-local Hamiltonian H acting on $n + (2s(n)T(n))^{1/c}$ qubits.

The construction is direct. By definition, g can be implemented by an $s(n)$-sized quantum circuit C^g. We can thus construct a circuit C concatenating $T(n)$ copies of C^g, which computes $f(T(n), x)$ with size $s(n)T(n)$. Note that, if we denote $C(i, x)$ to be the intermediate output of $C(x)$ after applying the i-th gate, then we additionally have $C(ks(n), x) = |f(k, x)\rangle$ for all $k \leq T(n)$.

Given such a circuit C, we can construct a Hamiltonian H by the circuit-to-Hamiltonian reduction introduced in Section 9.1. We use the construction in Lemma 48 with locality c and time-bound $2s(n)T(n)$ to implement the clock state. Hence $H = \sum_j H_j$, where $H_j := U_j \otimes |j\rangle\langle j-1| + U_j^\dagger \otimes |j-1\rangle\langle j|$, and U_j is a unitary corresponding to the j-th gate of C. Note that U_j is always an one- or two-qubit gate, and $|j\rangle\langle j-1|$ is c-local. Hence H is $c+2$ local over $n + (2s(n)T(n))^{1/c}$ qubits. It is also direct to see that $\|H_j\| = O(1)$ for each j.

For such H and any $t \in [0, s(n)T(n)]$, if there is a quantum algorithm $\mathcal{A}$ that computes e^{-iHt} within depth $d_{\mathcal{A}}$, we can indeed construct a quantum algorithm $\mathcal{R}$ with depth $d_{\mathcal{A}} + O(n)$ that computes the underlying parallel-hard puzzle. The algorithm $\mathcal{R}$ is defined as follows.

1. For an input $x \in \{0, 1\}^n$, run the algorithm $\mathcal{A}$ with input Hamiltonian H and input state $|x\rangle$, obtain the output state of $\mathcal{A}$, denoted by $|\psi\rangle$.
2. Measure the clock register $\mathcal{H}_{\text{clock}}$ in $\{|\gamma_j\rangle\}$ basis and obtain some $l \in [s(n)f(n)]$. The residual state in the circuit register $\mathcal{H}_{\text{circuit}}$ is denoted by $|\phi_l\rangle$.
3. Let $m = \lceil l/s(n) \rceil$. Apply $U_{ms(n)} \cdots U_{l+1}$ on $|\phi_l\rangle$. Let the final state be $|\phi_m\rangle$.
4. Measure $|\phi_m\rangle$ on the computational basis and obtain the outcome x_m.
5. Output x_m

In this construction, Step 2 and Step 4 can be done within depth $O(n)$ and Step 3 can be done within depth $s(n)$. It is easy to see that $\mathcal{R}$ can be implemented with depth $d_{\mathcal{A}} + O(s(n))$.

We claim that, with constant probability, $m > \frac{t}{s(n)}$ and $x_m = f(m, x)$. This implies that $\mathcal{R}$ can break the underlying parallel-hard puzzle on $k = 2\lfloor \frac{t}{s(n)} \rfloor$.

To prove the claim, we first consider a simplified case, where there exists an ideal $\widetilde{\mathcal{A}}$ that perfectly simulates H for time t and plugs it into our construction. We use $|\widetilde{\psi}\rangle$ to denote the output of $\widetilde{\mathcal{A}}$ in Step 1, and similarly $|\widetilde{\psi}_l\rangle$ for output in Step 2. By Lemma 47, we have $|\widetilde{\phi}_l\rangle = U_l \cdots U_1 |x\rangle$ for all l and the probability that we get some $l > t$ in Step 2 is at least $1/3$. By the definition of C, the output state of Step 3 satisfies $|\widetilde{\phi}_m\rangle = U_{ms(n)} \cdots U_1 |x\rangle = |f(m, x)\rangle$. Moreover, $m \geq \lfloor \frac{t}{s(n)} \rfloor$ with probability at least $1/3$. This matches the required condition of our claim.

Now we return to the general $\mathcal{A}$ with simulation error $\epsilon < 1/4$. Let $|\psi\rangle$ be the output of $\mathcal{A}$ with error ϵ. Observe that we have $\Delta(|\psi\rangle\langle\psi|, |\widetilde{\psi}\rangle\langle\widetilde{\psi}|) < 1/4$. Thus, by the definition of the trace distance, the difference in probabilities of obtaining any outcome by applying the same procedure to two states should be at most $1/4$. Hence, with probability at least $1/3 - 1/4 = 1/12$, measuring $|\phi_m\rangle$ gives $f(m, x)$ with some $m \geq \lfloor \frac{t}{s(n)} \rfloor$. This completes the proof of our claim.

Finally, by the security guarantee of the (s, d)-iterated parallel-hard puzzle, any circuit computing the puzzle for $k = 2\lfloor \frac{t}{s(n)} \rfloor$ should have depth at least $d(2\lfloor \frac{t}{s(n)} \rfloor)$. This gives an lower bound that $d_{\mathcal{A}} + O(s(n)) > d(2\lfloor \frac{t}{s(n)} \rfloor)$, which completes the proof.   ◀

9.3   Circuit to time-dependent Hamiltonian

In this section, we will show how to encode a circuit into a time-dependent geometrically local circuit.

▶ **Lemma 49.** *A quantum circuit C (of 2-qubit gates) over n qubits of size s can be transformed into a circuit C' over n qubits of size ns, such that every gate in C' acts only on consecutive qubits, and $C'(x) = \pi C(x)$ for some permutation π on n elements. We call such C' a geometrically local circuit.*

Proof. The proof of this small lemma is very direct. Given a circuit C with (sequential) gates $G_1, G_2, \ldots, G_s$, where gate G_i acts on qubits α_i, β_i, then we can rewrite C as $S_{1,\alpha_1}, S_{1,\alpha_1+1}, \ldots, S_{1,\beta_1-2}, G_1', G_2', \ldots, G_s'$, where S_i is a swap gate acting on the i and $(i+1)$-th qubits, and G_i' is G_i acting on the permuted qubits. Note that now G_1' is acting on two consecutive qubits $\beta_1 - 1, \beta_1$. It is easy to see that applying G_1 and applying $S_{1,\alpha_1}, S_{1,\alpha_1+1}, \ldots, S_{1,\beta_1-2}, G_1'$ generate output states that differ up to a permutation. Through repeating such process s times, we can obtain a circuit C' that consists of at most $(n-1)s$ swap gates and s permuted gates from C. Furthermore, every gate in C' is acting only on consecutive qubits.   ◀

▶ **Theorem 50.** *Given a quantum circuit C over n qubits that consists of s gates $U_1 \ldots U_s$, we can define a time-dependent Hamiltonian H such that $H(t) := -i \log U_i$ for all $t \in [i-1, i)$ and $U_i := I_n$ for $i > s$. Note that H obviously satisfies $e^{iHt}|\phi\rangle = C|\phi\rangle$ for $t > s$.*

▶ Remark 51. A Hamiltonian H obtained from a geometrically local circuit is 1-D geometrically 2-local.

▶ Remark 52. Such time-independent Hamiltonians that remain constant in each time segment are also called *piecewise constant Hamiltonians* in [23].

9.4 Proof of the lower bound for geometrically local Hamiltonians

Before proving the theorem, we will need the existence of a Hamiltonian simulation algorithm for 2-local *time-independent* Hamiltonians. While there are plenty of proposals in the literature, we use the one in [39] which provides a good dependency on ϵ. The choice of the simulation algorithm will only affect an additive term of our bound.

▶ **Theorem 53** ([39], with some parameter specified). *A 2-local Hamiltonian acting on n qubits can be simulated for time t within precision ϵ by a quantum circuit of depth $\text{poly}(\log\log(1/\epsilon), \log(n), t)$.*

Proof of Theorem 46. The proof is very similar to the proof of Theorem 45. From an (s, d)-iterated parallel-hard function $f(k, x) = g^{(k)}(x)$, we will construct a geometrically 2-local Hamiltonian H over n qubits.

Again, since g can be implemented by an $s(n)$ sized circuit C^g, we can construct C^f by concatenating $T(n)$ copies of C^g, which computes $f(T(n), x)$ with size $s(n)T(n)$. Then, by applying Lemma 49, we can obtain a geometrically local circuit C of depth at most $ns(n)T(n)$. Note that Lemma 49 basically constructs C by adding at most $n - 1$ swap gates before each gate of C^f. For notation simplicity, we add extra dummy gate before each gate so that there are exactly $(n - 1)$ gates added before each gate of C^f. Thus the depth of C is exactly $ns(n)T(n)$, and if we denote $C(i, x)$ to be the intermediate output of $C(x)$ after applying the i-th gate, we have $C(kns(n), x) = \pi_k|f(k, x)\rangle$ for all $k \leq T(n)$, where π_k is some (known) permutation on n bits.

With such a geometrically local circuit C, we can apply Theorem 50 to obtain a time-dependent geometrically 2-local Hamiltonian H over n qubits. For all $t \in [0, ns(n)T(n)]$, if there is a quantum algorithm $\mathcal{A}$ that computes e^{-iHt} with precision ϵ of depth $d_{\mathcal{A}}$, we can indeed construct a quantum algorithm $\mathcal{R}$ of depth $d_{\mathcal{A}} + O(s(n)) + \text{poly}(\log\log(1/\epsilon'), \log n)$ with $\epsilon' < 1 - \epsilon - 1/\text{poly}(n)$. The algorithm $\mathcal{R}$ is defined as follows.

1. For an input $x \in \{0, 1\}^n$, run $\mathcal{A}$ on inputs the Hamiltonian H and the initial state $|x\rangle$ to obtain the output state $|\phi\rangle$.
2. Run the Hamiltonian simulation algorithm $\mathcal{S}$ in Theorem 53 with inputs the Hamiltonian $H(t)$ (which is time-independent) , the initial state $|\phi\rangle$, the evolution time $\lceil t \rceil - t$, and the precision parameter ϵ' to obtain the output state $|\phi_{\lceil t \rceil}\rangle$.
3. Let $m = \lceil \frac{t}{ns(n)} \rceil$. Apply $U_{mns(n)} \cdot U_{\lceil t \rceil + 1}$ to $|\phi_{\lceil t \rceil}\rangle$. Let the final state be $|\phi_m\rangle$.
4. Measure the permuted state $\pi_m|\phi_m\rangle$ on the computational basis and obtain the outcome x_m.
5. Output x_m.

In the construction, Step 3 can be done within depth $O(ns(n))$ and Step 4 can be done within depth $O(1)$. Thus, R can be instantiated within depth $d_{\mathcal{A}} + O(ns(n)) + \text{poly}(\log\log(1/\epsilon'), \log n)$.

Now we show that if $\epsilon + \epsilon' \leq 1 - 1/\operatorname{poly}(n)$, then $x_m = f(k,m)$ holds with non-negligible probability.

We consider the case in which the algorithms $\mathcal{A}$ and $\mathcal{S}$ both simulate the evolution perfectly. Denote the output state in each step of this experiment as $|\tilde{\phi}\rangle, |\tilde{\phi}_{\lceil t \rceil}\rangle$ and $|\tilde{\phi}_m\rangle$. In particular,

$$|\tilde{\phi}\rangle = \exp_{\mathcal{T}}\left(-i \int_0^t H(t')dt'\right)|x\rangle$$

and

$$|\tilde{\phi}_{\lceil t \rceil}\rangle = e^{-iH(t)(\lceil t \rceil - t)} \times \exp_{\mathcal{T}}\left(-i \int_0^t H(t')dt'\right)|x\rangle$$

$$= \exp_{\mathcal{T}}\left(-i \int_0^{\lceil t \rceil} H(t')dt'\right)|x\rangle = |C(\lceil t \rceil, x)\rangle,$$

where the last equation follows from the definition of H, while the second last equation follows from the fact that H is constant on the interval $(t, \lceil t \rceil)$. Hence, $|\tilde{\phi}_m\rangle = |C(mns(n), x)\rangle = |f(m,x)\rangle$ with probability 1.

Back to the actual construction, by the precision guarantee of $\mathcal{A}$ and $\mathcal{S}$, there exists some polynomial p such that $\Delta(|\psi_{\lceil t \rceil}\rangle\langle\psi_{\lceil t \rceil}|, |\tilde{\psi}_{\lceil t \rceil}\rangle\langle\tilde{\psi}_{\lceil t \rceil}|) < 1 - 1/p(n)$. Thus, $x_m = f(m,x)$ holds with probability at least $1/p(n)$.

Finally, by the security guarantee of the (s,d)-iterated parallel-hard function, any quantum circuit that computes $f(m,x)$ should have depth at least $d(m) = d(\lceil \frac{t}{ns(n)} \rceil)$. This gives an lower bound that $d_{\mathcal{A}} + O(ns(n)) + \operatorname{poly}(\log\log(1/\epsilon), \log(n)) < d(\lceil \frac{t}{ns(n)} \rceil)$, which completes the proof. ◄

10   No Fast-forwarding with Natural Simulators

In previous sections, we have shown no-go theorems for using quantum circuits to parallelly fast-forward (geometrically) local Hamiltonian simulation. Here, we are going to generalize Theorem 46 and Theorem 45 by showing that simulators that are geometrically local Hamiltonians cannot do much better than quantum circuits.

▶ **Corollary 54.** *Assuming an (s,d)-iterated parallel-hard function, for any integer n, there exists a piecewise-time-independent 1-D geometrically 2-local Hamiltonian H_A acting on n qubits satisfying the following: For any geometrically constant-local Hamiltonian H_B acting on $\operatorname{poly}(n)$ qubits, using H_B to simulate H_A for evolution time $t \in [0, ns(n)T(n)]$ with error $\epsilon(n) \leq 1 - 1/\operatorname{poly}(n)$ needs $(d(\lfloor \frac{t}{ns(n)} \rfloor) - O(ns(n)) - \operatorname{poly}(\log(n), \log\log(1/\epsilon'(n)))/\operatorname{polylog}(tn/\epsilon)$ evolution time, where $T(\cdot)$ is an arbitrary polynomial and $\epsilon'(n) < 1 - \epsilon(n) - 1/\operatorname{poly}(n)$.*

Proof. Let H_A be the Hamiltonian we considered in the proof of Theorem 46. Suppose there exists H_B that can simulate H_A for evolution time $t \in [0, ns(n)T(n)]$ and error $\epsilon(n)/2 \leq 1 - 1/\operatorname{poly}(n)$ with $t' < \frac{d(\lfloor \frac{t}{ns(n)} \rfloor) - O(ns(n)) - \operatorname{poly}(\log(n), \log\log(1/\epsilon'(n)))}{\operatorname{polylog}(tn/\epsilon)}$ evolution time, where $\epsilon'(n) < 1 - \epsilon(n)/2 - 1/\operatorname{poly}(n)$.

Recall that the algorithm in [23] can simulate a geometrically constant-local Hamiltonian with quantum circuit depth $t \cdot \operatorname{polylog}(tn/\epsilon)$, where n is the number of qubits and ϵ is the precision parameter.

Then, we apply the algorithm in [23] to simulate H_B with evolution time t' and precision $\epsilon/2$. This leads to a simulation algorithm for H_A with error ϵ and circuit depth strictly less than $d(\lfloor \frac{t}{ns(n)} \rfloor) - O(ns(n)) - \operatorname{poly}(\log(n), \log\log(1/\epsilon'(n)))$, which contradicts Theorem 46. ◄

▶ **Corollary 55.** *Assuming an (s, d)-iterated parallel-hard function, for any integer n, there exists a time-independent c-local Hamiltonian H acting on $n + (2s(n)T(n))^{1/c}$ qubits satisfying the following: For any geometrically constant-local Hamiltonian H_B acting on* $\mathrm{poly}(n)$ *qubits, using H_B to simulate H_A for evolution time $t \in [0, s(n)T(n)]$ with error $\epsilon(n) < 1/4$ needs $d(\lfloor t/2s(n) \rfloor) - O(s(n))/\mathrm{polylog}(tn/\epsilon)$ evolution time, where $T(\cdot)$ is an arbitrary polynomial and c is a constant.*

The proof for Corollary 55 is similar to the proof for Corollary 54.

─── **References** ───

1   Dorit Aharonov, Wim van Dam, Julia Kempe, Zeph Landau, Seth Lloyd, and Oded Regev. Adiabatic Quantum Computation Is Equivalent to Standard Quantum Computation. *SIAM Review*, 50(4):755–787, 2008.

2   Brian Alspach. Johnson graphs are Hamilton-connected. *Ars Mathematica Contemporanea*, 6(1):21–23, 2012.

3   Yosi Atia and Dorit Aharonov. Fast-forwarding of hamiltonians and exponentially precise measurements. *Nature Communications*, 8(1), November 2017. `doi:10.1038/s41467-017-01637-7`.

4   Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *Proceedings of the 1st ACM Conference on Computer and Communications Security*, pages 62–73, 1993.

5   Charles H Bennett, Ethan Bernstein, Gilles Brassard, and Umesh Vazirani. Strengths and weaknesses of quantum computing. *SIAM journal on Computing*, 26(5):1510–1523, 1997.

6   Dominic W Berry, Graeme Ahokas, Richard Cleve, and Barry C Sanders. Efficient quantum algorithms for simulating sparse Hamiltonians. *Communications in Mathematical Physics*, 270(2):359–371, 2007.

7   Dominic W. Berry, Andrew M. Childs, Richard Cleve, Robin Kothari, and Rolando D. Somma. Exponential improvement in precision for simulating sparse Hamiltonians. *Proceedings of the forty-sixth annual ACM symposium on Theory of computing*, pages 283–292, May 2014. arXiv: 1312.1414. `doi:10.1145/2591796.2591854`.

8   Dominic W. Berry, Andrew M. Childs, and Robin Kothari. Hamiltonian Simulation with Nearly Optimal Dependence on all Parameters. In *2015 IEEE 56th Annual Symposium on Foundations of Computer Science*, pages 792–809, October 2015. ISSN: 0272-5428. `doi:10.1109/FOCS.2015.54`.

9   Dan Boneh, Joseph Bonneau, Benedikt Bünz, and Ben Fisch. Verifiable delay functions. In *CRYPTO (1)*, volume 10991 of *Lecture Notes in Computer Science*, pages 757–788. Springer, 2018.

10   Dan Boneh, Özgür Dagdelen, Marc Fischlin, Anja Lehmann, Christian Schaffner, and Mark Zhandry. Random oracles in a quantum world. In *Advances in Cryptology–ASIACRYPT 2011: 17th International Conference on the Theory and Application of Cryptology and Information Security, Seoul, South Korea, December 4-8, 2011. Proceedings 17*, pages 41–69. Springer, 2011.

11   P Oscar Boykin, Tal Mor, Matthew Pulver, Vwani Roychowdhury, and Farrokh Vatan. On universal and fault-tolerant quantum computing: a novel basis and a new constructive proof of universality for Shor's basis. In *40th Annual Symposium on Foundations of Computer Science (Cat. No. 99CB37039)*, pages 486–494. IEEE, 1999.

12   Jorge Chávez-Saab, Francisco Rodríguez-Henríquez, and Mehdi Tibouchi. Verifiable isogeny walks: Towards an isogeny-based postquantum VDF. In *SAC*, volume 13203 of *Lecture Notes in Computer Science*, pages 441–460. Springer, 2021.

13   Andrew M. Childs and Dominic W. Berry. Black-box Hamiltonian simulation and unitary implementation. *Quantum Information and Computation*, 12(1&2):29–62, January 2012. `doi:10.26421/QIC12.1-2-4`.

14   Andrew M. Childs, Richard Cleve, Enrico Deotto, Edward Farhi, Sam Gutmann, and Daniel A. Spielman. Exponential algorithmic speedup by a quantum walk. In *Proceedings of the thirty-fifth ACM symposium on Theory of computing - STOC '03*. ACM Press, 2003. `doi:10.1145/780542.780552`.

15   Andrew MacGregor Childs. *Quantum Information Processing in Continuous Time*. Thesis, Massachusetts Institute of Technology, 2004.

16   Kai-Min Chung, Serge Fehr, Yu-Hsuan Huang, and Tai-Ning Liao. On the compressed-oracle technique, and post-quantum security of proofs of sequential work. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 598–629. Springer, 2021.

17   J. Ignacio Cirac and Peter Zoller. Goals and opportunities in quantum simulation. *Nature Physics*, 8(4):264–266, April 2012. `doi:10.1038/nphys2275`.

18   Toby Cubitt, Ashley Montanaro, and Stephen Piddock. Universal Quantum Hamiltonians. *Proceedings of the National Academy of Sciences*, 115(38):9497–9502, September 2018. `doi:10.1073/pnas.1804949115`.

19   Naomi Ephraim, Cody Freitag, Ilan Komargodski, and Rafael Pass. Continuous verifiable delay functions. In *EUROCRYPT (3)*, volume 12107 of *Lecture Notes in Computer Science*, pages 125–154. Springer, 2020.

20   Luca De Feo, Simon Masson, Christophe Petit, and Antonio Sanso. Verifiable delay functions from supersingular isogenies and pairings. In *ASIACRYPT (1)*, volume 11921 of *Lecture Notes in Computer Science*, pages 248–277. Springer, 2019.

21   Richard P. Feynman. Quantum Mechanical Computers. *Optics News*, 11(2):11–20, February 1985. `doi:10.1364/ON.11.2.000011`.

22   Shouzhen Gu, Rolando D. Somma, and Burak Ş ahinoğlu. Fast-forwarding quantum evolution. *Quantum*, 5:577, November 2021. `doi:10.22331/q-2021-11-15-577`.

23   Jeongwan Haah, Matthew B. Hastings, Robin Kothari, and Guang Hao Low. Quantum algorithm for simulating real time evolution of lattice Hamiltonians. *SIAM Journal on Computing*, pages FOCS18–250–FOCS18–284, January 2021. `doi:10.1137/18m1231511`.

24   Stacey Jeffery, Frederic Magniez, and Ronald de Wolf. Optimal parallel quantum query algorithms, 2013. `doi:10.48550/ARXIV.1309.6116`.

25   A. Kitaev, A. Shen, and M. Vyalyi. *Classical and Quantum Computation*, volume 47 of *Graduate Studies in Mathematics*. American Mathematical Society, Providence, Rhode Island, May 2002. `doi:10.1090/gsm/047`.

26   Ilia Krasikov. Uniform bounds for Bessel functions. *Journal of Applied Analysis*, 12(1):83–91, 2006.

27   Guang Hao Low and Isaac L. Chuang. Optimal Hamiltonian simulation by quantum signal processing. *Phys. Rev. Lett.*, 118:010501, January 2017. `doi:10.1103/PhysRevLett.118.010501`.

28   Guang Hao Low and Isaac L. Chuang. Hamiltonian Simulation by Qubitization. *Quantum*, 3:163, July 2019. `doi:10.22331/q-2019-07-12-163`.

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

30   Daniel Nagaj. Fast universal quantum computation with railroad-switch local Hamiltonians. *Journal of Mathematical Physics*, 51(6):062201, 2010.

31   Michael A. Nielsen and Isaac L. Chuang. *Quantum Computation and Quantum Information: 10th Anniversary Edition*. Cambridge University Press, 2010. `doi:10.1017/CBO9780511976667`.

32   Krzysztof Pietrzak. Simple verifiable delay functions. In *ITCS*, volume 124 of *LIPIcs*, pages 60:1–60:15. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019.

33   Ronald L Rivest, Adi Shamir, and David A Wagner. Time-lock puzzles and timed-release crypto. *Massachusetts Institute of Technology. Laboratory for Computer Science*, 1996.

34   J. J. Sakurai and Jim Napolitano. *Modern Quantum Mechanics*. Cambridge University Press, third edition, September 2020. `doi:10.1017/9781108587280`.

35 Benjamin Wesolowski. Efficient verifiable delay functions. *J. Cryptol.*, 33(4):2113–2147, 2020.

36 Christof Zalka. Grover's quantum searching algorithm is optimal. *Physical Review A*, 60(4):2746, 1999.

37 Mark Zhandry. A note on the quantum collision and set equality problems, 2013. `doi: 10.48550/arXiv.1312.1027`.

38 Mark Zhandry. How to record quantum queries, and applications to quantum indifferentiability. In *Annual International Cryptology Conference*, pages 239–268. Springer, 2019.

39 Zhicheng Zhang, Qisheng Wang, and Mingsheng Ying. Parallel quantum algorithm for Hamiltonian simulation. *arXiv preprint*, 2021. `arXiv:2105.11889`.