Quantum Algorithms on Edge Lists: Hiding, Shuffling, and Cycle Finding

Amin Shiraz Gilani
University of Maryland, College Park, MD, USA

Daochen Wang
University of British Columbia, Vancouver, Canada

Pei Wu
The Pennsylvania State University, University Park, PA, USA

Xingyu Zhou
University of British Columbia, Vancouver, Canada

Abstract

The edge list model is arguably the simplest input model for graphs, where the graph is specified by a list of its edges. In this model, we study the quantum query complexity of three variants of the triangle finding problem. The first asks whether there exists a triangle containing a target edge and raises general questions about the hiding of a problem's input among irrelevant data. The second asks whether there exists a triangle containing a target vertex and raises general questions about the shuffling of a problem's input. The third asks whether there exists a triangle; this problem bridges the 3-distinctness and 3-sum problems, which have been extensively studied by both cryptographers and complexity theorists. We provide tight or nearly tight results for these problems as well as some first answers to the general questions they raise.

Furthermore, given any graph with low maximum degree, such as a typical random sparse graph, we prove that the quantum query complexity of finding a length-k cycle in its length-m edge list is $m^{3/4-1/(2^{k+2}-4)\pm o(1)}$, which matches the best-known upper bound for the quantum query complexity of k-distinctness on length-m inputs up to an $m^{o(1)}$ factor. We prove the lower bound by developing new techniques within Zhandry's recording query framework [43] as generalized by Hamoudi and Magniez [23]. These techniques extend the framework to treat any non-product distribution that results from conditioning a product distribution on the absence of rare events. We prove the upper bound by adapting Belovs's learning graph algorithm for k-distinctness [11]. Finally, assuming a plausible conjecture concerning only cycle finding, we show that the lower bound can be lifted to an essentially tight lower bound on the quantum query complexity of k-distinctness, which is a long-standing open question.

2012 ACM Subject Classification Theory of computation → Quantum query complexity

Keywords and phrases Quantum query complexity, graph algorithms, edge list model

Digital Object Identifier 10.4230/LIPIcs.ICALP.2026.97

Category Track A: Algorithms, Complexity and Games

Related Version *Full Version*: `https://arxiv.org/abs/2412.17786` [22]

Funding *Amin Shiraz Gilani*: DOE ASCR Quantum Testbed Pathfinder program Grants DE-SC0019040 and DE-SC0024220.
Daochen Wang: NSERC Grants CRC-2023-00039, RGPIN-2024-06493, and DGECR-2024-00113.
Xingyu Zhou: NSERC Grant RGPIN-2024-06493

Acknowledgements We thank François Le Gall for suggesting the study of quantum algorithms in the edge list model. We thank Yassine Hamoudi for helpful discussions on [23] and for simplifying our original proof of the Mirroring Lemma in the binary-alphabet case. We thank anonymous reviewers for identifying errors in the Mirroring Lemma in earlier versions of this paper.

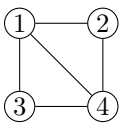
53rd International Colloquium on Automata, Languages, and Programming (ICALP 2026).
Editors: Sayan Bhattacharya, Danupon Nanongkai, Michael Benedikt, and Gabriele Puppis;
Article No. 97; pp. 97:1–97:16

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

1 Introduction

The study of graph problems forms a cornerstone of research in theoretical computer science. These problems have been studied when the input data structure is the *adjacency matrix* or *adjacency list* of the graph. In the adjacency matrix model of an n-vertex undirected simple graph, one is given the presence or absence status of every one of the $\binom{n}{2}$ edges in an $n \times n$ binary matrix. In the adjacency list model, the input is given as n lists of neighbors, one for each of the n vertices. Both models build structural cues into the input: an adjacency matrix assigns each vertex a given row and column, while an adjacency list uses vertices to order edges. These cues allow for fast computation but are arguably not intrinsic to the underlying graph.

In this work, we study graph problems in the edge list model, a minimalist model where the graph is given as an unordered multiset of edges. When computing on edge lists, algorithms must build the graph's structure for themselves and not rely on any input cues. Thus, studying computation on edge lists offers a new window into the relationship between a graph problem's *structure* and *complexity*.



Edge list: {1,2} {3,4} {1,4} {1,3} {2,4}

Adjacency list: 1: 2 4 3 2: 4 1 3: 1 4 4: 1 3 2

Figure 1 An edge list can be seen as a shuffled adjacency list: both inputs specify the same displayed graph.

We study the edge list model in the context of *quantum query complexity*. In quantum query complexity, we assume access to a quantum computer and characterize the complexity of a problem by the number of times the computer queries an input edge list.

Quantum algorithms on edge lists has previously been studied in the area of streaming algorithms. For example, [28] showed that triangle counting can be solved by a quantum streaming algorithm using less space than any classical rival. However, the quantum computational model employed differs significantly from the quantum query model that we study. In the former case, the edge list is still queried classically – only the memory is quantum – whereas we assume the edge list can be queried in *quantum superposition*. As we will see in this work, this change of model leads to new and interesting questions.

Inspired by the fact that the triangle problem in the adjacency matrix model has driven innovations in quantum query complexity [34, 10, 25, 30, 29, 42] over many years, we were driven to study the same problem but in the edge list model. Specifically, we study three variants of the triangle problem.

- TriangleEdge asks whether there exists a triangle containing a target edge and raises general questions about how a problem's complexity increases if its input is *hidden* among irrelevant data. For the specific problem, the irrelevant data consists of those edges in the input that are not incident to the target edge.
- TriangleVertex asks whether there exists a triangle containing a target vertex and raises general questions about how a problem's complexity increases if its input is *shuffled*. For the specific problem, two parts of the input are shuffled: one part containing edges incident to the target vertex and the other part containing the remaining edges. The notion of shuffling is also inherent in the comparison between the edge list and adjacency list models.

- Triangle asks whether a triangle exists and bridges the 3-distinctness and 3-sum problems. The latter two problems have been extensively studied, both in the worst-case setting by complexity theorists seeking to connect problem structure with algorithmic complexity [11, 15, 13, 14, 6] and in the average-case setting by cryptographers seeking to understand the security of cryptosystems [41, 32].

We provide tight or nearly tight results for these problems as well as some first answers to the general questions they raise. Furthermore, we generalize our results for Triangle to k-cycle finding, a problem motivated by its close ties to the well-studied problem of k-distinctness.

2 Our results

In the following, we write $Q(\cdot)$ for the worst-case (bounded-error) quantum query complexity, $\mathbb{N}$ for the set of positive integers, and $[n] \coloneqq \{1, \ldots, n\}$ for every $n \in \mathbb{N}$.

2.1 TriangleEdge and Hiding

In TriangleEdge, when searching for a triangle containing a target edge $\{u, v\}$, any edge in the input of the form $\{u', v'\}$ such that $\{u', v'\} \cap \{u, v\} = \emptyset$ can be viewed as irrelevant data that hides the relevant part of the input. This observation motivates a general definition of hiding.

▶ **Definition 1** (Hiding transform). *Let a, b be integers with $b \geq a \geq 1$. Let $\widetilde{D}, \Sigma$ be finite non-empty sets with $\widetilde{D} \subseteq \Sigma^a$. For a function $f\colon \widetilde{D} \to \{0,1\}$, we define the* hiding transform *of f to be*

$$\mathsf{HIDE}_b[f]\colon D \subseteq (\Sigma \cup \{*\})^b \to \{0,1\}, \tag{1}$$

where $*$ is a symbol outside of Σ, the set D consists of all strings $y \in (\Sigma \cup \{*\})^b$ with exactly a non-$*$ symbols such that the length-a subsequence[1] $\tilde{y}$ formed by those non-$*$ symbols is an element of $\widetilde{D}$, and $\mathsf{HIDE}_b[f](y)$ is defined to be $f(\tilde{y})$.

Then TriangleEdge can be seen as $\mathsf{HIDE}_m[\mathrm{ED}_d]$, where ED_d is the element distinctness function that decides whether an input string of length d has two distinct positions containing the same symbol. We show the following.[2]

▶ **Theorem 2** (Informal statement of Theorem 3.5 and Corollary 3.7 of the full version). *$Q(\mathsf{HIDE}_m[\mathrm{ED}_d]) = \widetilde{\Theta}(\sqrt{m}d^{1/6})$. Consequently,*

$$Q(\mathsf{TriangleEdge}) = \widetilde{\Theta}(\sqrt{m}d^{1/6}), \tag{2}$$

where the input is a length-m edge list and the target edge has d neighboring edges.[3]

The study of TriangleEdge naturally led us to investigate how $Q(f)$ relates to $Q(\mathsf{HIDE}_b[f])$ for arbitrary $f\colon \Sigma^a \to \{0,1\}$. As a first step in this direction, we show $Q(\mathsf{HIDE}_b[f]) = \widetilde{\Theta}(\sqrt{b/a} \cdot Q(f))$ for every symmetric $f\colon \{0,1\}^a \to \{0,1\}$, where "symmetric" refers to "symmetric under permuting the *positions* of input symbols" throughout this work. The

[1] In general, for integers $b \geq a \geq 1$, Σ a finite non-empty set, and a string $x \in \Sigma^b$: a length-a subsequence of x is a string of the form $x_{i_1}x_{i_2}\ldots x_{i_a}$ for some integers $1 \leq i_1 < i_2 < \cdots < i_a \leq b$.

[2] We use the notation $\widetilde{O}, \widetilde{\Omega}, \widetilde{\Theta}$ to denote big-O,Ω, and Θ up to poly-logarithmic factors.

[3] If d is not given, it can be estimated to within a constant factor by approximate counting at a lower asymptotic cost than the stated upper bound.

symmetry condition is necessary. For example, let $f : \{0,1\}^a \to \{0,1\}$ be the dictator function $f(x) = x_1$. Then $Q(f) = 1$, but $\mathsf{HIDE}_b[f]$ requires finding the first non-$*$ symbol among the b positions. On the subdomain where the last $a-1$ positions are fixed non-$*$ symbols and exactly one of the first $b-a+1$ positions is non-$*$, computing $\mathsf{HIDE}_b[f]$ contains unstructured search on $b-a+1$ items as a subproblem. Hence

$$Q(\mathsf{HIDE}_b[f]) = \Omega(\sqrt{b-a+1}), \tag{3}$$

which is not $O(\sqrt{b/a}\,Q(f))$ in general. On the other hand, we conjecture that the result holds for every symmetric $f\colon \Sigma^a \to \{0,1\}$, even if $|\Sigma| > 2$. We support this conjecture by showing that its randomized analogue is true in Proposition 3.10 of the full version.

2.2 TriangleVertex and Shuffling

We show in Propositions 4.1 and 4.2 of the full version that the quantum query complexity of TriangleVertex, when the input is a length-m edge list and the target vertex u has degree d, which need not be known in advance, satisfies

$$\Omega(\sqrt{m/d} \cdot Q(3\text{-}\mathsf{DIST}_d)) \leq Q(\mathsf{TriangleVertex}) \leq O(\sqrt{m}d^{1/4}), \tag{4}$$

where $3\text{-}\mathsf{DIST}_d$ is the 3-distinctness function that decides whether an input string of length d has three distinct positions containing the same symbol. Note that $Q(3\text{-}\mathsf{DIST}_d)$ is between $\Omega(d^{2/3})$ – because it is no easier than ED_d (which could be also called $2\text{-}\mathsf{DIST}_d$) – and $O(d^{5/7})$ [11].

When seeking a triangle containing the target vertex u, it is helpful to think of the input edge list as consisting of two parts: part A containing edges that are incident to u, and part B containing edges that are not incident to u. We do not a priori know where parts A and B are but neither part contains irrelevant data – indeed two edges of the triangle must come from A and one edge from B (if it exists). The *shuffling* of parts A and B contributes to the hardness of TriangleVertex.

The above discussion and the comparison of the edge list model with the adjacency list and adjacency matrix models naturally led us to investigate how shuffling a function's input affects its complexity. We formalize and study two concrete versions of this problem.

2.2.1 Shuffled functions

As discussed, the edge list can be seen as a "shuffled version" of the adjacency list. We now formalize this notion.

▶ **Definition 3** (Shuffling transform). *Let $n \in \mathbb{N}$. Let $\widetilde{D}, \Sigma$ be finite non-empty sets with $\widetilde{D} \subseteq \Sigma^n$. For a function $f\colon \widetilde{D} \to \{0,1\}$, we define the* shuffling transform *of f to be*

$$\mathsf{SHUFFLE}[f]\colon D \subseteq (\Sigma \times [n])^n \to \{0,1\}, \quad \textit{where} \tag{5}$$

1. *the set D consists of all elements $x \in (\Sigma \times [n])^n$ satisfying $x = ((v_1, \pi(1)), \ldots, (v_n, \pi(n)))$ for some bijection $\pi\colon [n] \to [n]$ such that $(v_{\pi^{-1}(1)}, v_{\pi^{-1}(2)}, \ldots, v_{\pi^{-1}(n)}) \in \widetilde{D}$.*
2. $\mathsf{SHUFFLE}[f](x) \coloneqq f(v_{\pi^{-1}(1)}, v_{\pi^{-1}(2)}, \ldots, v_{\pi^{-1}(n)})$.

In other words, the inputs to $\mathsf{SHUFFLE}[f]$ are shuffled versions of the inputs to f such that each symbol of the input to $\mathsf{SHUFFLE}[f]$ additionally contains its position pre-shuffling. Of course, every (worst-case) query complexity measure of $\mathsf{SHUFFLE}[f]$ is at least that of f, since $\mathsf{SHUFFLE}[f]$ contains a copy of f under the restriction of the domain D to those inputs

of the form $(x_1, 1), \ldots, (x_n, n)$, where $x = x_1 \ldots x_n \in \widetilde{D}$. If f is symmetric, then every query complexity measure on f is the same as that measure on $\mathsf{SHUFFLE}[f]$ since the algorithm computing $\mathsf{SHUFFLE}[f]$ could simply ignore the second coordinates of the input. If f is not symmetric, then $Q(\mathsf{SHUFFLE}[f])$ could be much larger than $Q(f)$: the dictator function witnesses a $\Omega(\sqrt{n})$ versus 1 separation.

Given the above discussion, the interesting question becomes: how does the separation between $Q(\mathsf{SHUFFLE}[f])$ and $Q(f)$ depend on "how symmetric" f is? Natural (partially) symmetric f arises from graph properties. We show that there can be massive separations between $Q(\mathsf{SHUFFLE}[f])$ and $Q(f)$ even if f has significant symmetry by being defined as a graph property in either (i) the adjacency list model (exponential separation) or (ii) adjacency matrix model (unbounded separation).

Since an edge list can be interpreted as a shuffled adjacency list, result (i) can be interpreted as "quantum computers can compute some graph property exponentially faster given an adjacency list instead of an edge list"; the proof of result (ii) can also be adapted to show "quantum computers can compute some graph property unboundedly faster given an adjacency matrix instead of an edge list".

▶ **Theorem 4** (Informal statement of Theorems 4.5 and 4.6 of the full version)**.**

1. *There exists a (family of) graph property $\mathcal{P}_1 : A \to \{0,1\}$, where A denotes a set of adjacency lists of size n such that*

$$Q(\mathcal{P}_1) = O(\mathrm{polylog}(n)) \quad \textit{and} \quad Q(\mathsf{SHUFFLE}[\mathcal{P}_1]) = n^{\Omega(1)}.$$

2. *There exists a (family of) graph property $\mathcal{P}_2 : M \to \{0,1\}$, where M denotes a set of $n \times n$ adjacency matrices such that*

$$Q(\mathcal{P}_2) = O(1) \quad \textit{and} \quad Q(\mathsf{SHUFFLE}[\mathcal{P}_2]) = n^{\Omega(1)}.$$

2.2.2 Shuffled direct sum

In TriangleVertex, the hardness arising from shuffling is intuitively not due to shuffling *within* parts A and B but rather *between* them. We formalize a toy version of this type of shuffling in the context of direct sum as follows.

▶ **Definition 5** (Shuffled direct sum)**.** *Let $n, k \in \mathbb{N}$. Let Σ be a finite non-empty set. For a function $f\colon \Sigma^n \to \{0,1\}$, we define the k-shuffled direct sum of f to be*

$$\mathsf{SHUFFLE}^k[f]\colon D \subseteq (\Sigma \times [k])^{kn} \to \{0,1\}^k, \quad \textit{where} \tag{6}$$

1. *the set D consists of all elements $x \in (\Sigma \times [k])^{kn}$ satisfying $x = ((v_1, c_1), \ldots, (v_{kn}, c_{kn}))$ such that: for all $j \in [k]$, there are exactly n indices $i \in [kn]$ with $c_i = j$.*
2. *$\mathsf{SHUFFLE}^k[f](x)$ is defined to equal $(f(v^{(1)}), \ldots, f(v^{(k)}))$, where $v^{(j)}$ is the subsequence of $v := v_1 \ldots v_{kn}$ indexed by those $i \in [kn]$ such that $c_i = j$.*

We have $Q(\mathsf{SHUFFLE}^k[f]) \geq \Omega(kQ(f))$, since computing $\mathsf{SHUFFLE}^k[f]$ is at least as hard as computing k independent copies of f, which costs $\Omega(kQ(f))$ queries by a well-known direct-sum theorem for quantum query complexity [4, 38]. This direct sum lower bound can be far from tight for $\mathsf{SHUFFLE}^k[f]$: again consider f being the dictator function. At first sight, computing $\mathsf{SHUFFLE}^k[f]$ seems harder than computing the k independent copies of f even if f is symmetric. Perhaps counterintuitively, we show in Proposition 4.9 of the full version that this is not the case if f is symmetric and has Boolean domain $\{0,1\}^n$: for such f, we show $Q(\mathsf{SHUFFLE}^k[f]) \leq O(kQ(f))$. In Conjecture 4.10 of the full version, we conjecture that the same holds for symmetric f with non-Boolean domain.

2.3 Triangle Finding

Having discussed TriangleEdge and TriangleVertex, we now turn to Triangle, where we simply ask whether a triangle exists[4] without placing any constraints. In the edge list model, we find that Triangle bridges the 3-DIST and 3-SUM problems since its structure lies between theirs. Recall that in 3-DIST, the goal is to decide whether the input contains three repetitions of the same symbol. In 3-SUM, the input contains symbols from some abelian group and the goal is to decide if there are three symbols that sum to the zero-element of the group.

Intuitively, Triangle should be no easier than 3-DIST and no harder than 3-SUM by considering its "certificate structure" [14]. In 3-DIST, once we have found one symbol of a 1-certificate, the next symbol for completing the certificate is determined. In 3-SUM, once we have found one symbol of a 1-certificate, the next symbol can be arbitrary. In comparison, in Triangle, once we have found one edge (i, j) of a 1-certificate, the next edge must be incident to one of i or j so it is neither determined nor arbitrary. In Proposition 5.1 of the full version, we show that this intuition is formally correct by reducing 3-DIST to Triangle and Triangle to 3-SUM.

Our first main result is stated informally below.

▶ **Theorem 6** (Informal statement of Theorem 5.5 and Corollary 5.6 of the full version)**.** *Given a length-m edge list on $n = \Theta(m)$ vertices, the quantum query complexity of finding a triangle is*

$$Q(\mathsf{Triangle}) \geq \widetilde{\Omega}(m^{5/7}). \tag{7}$$

Furthermore, the lower bound holds even if the edge list is uniformly random, which corresponds to a random sparse graph. As a corollary, Equation (7) *holds even if the edge list is promised to have a maximum degree at most $O(\log(m)/\log\log(m))$.*

We find Theorem 6 interesting for several reasons. Firstly, proving the theorem pushed us to develop new techniques for the recording query framework, which we believe to be of independent interest. The recording query framework was pioneered by Zhandry [43] to prove the security of cryptosystems against quantum adversaries. There is now a growing line of work developing new techniques within the recording query framework, including [32, 21, 23, 9, 35, 33, 19]. Of particular relevance to our work is [23] by Hamoudi and Magniez, which generalized Zhandry's original framework in order to lower bound the quantum query complexity of finding collision pairs.

This framework is particularly well suited for proving average-case quantum query lower bounds, where the input is sampled from certain types of distributions. The framework can also yield *optimal* worst-case lower bounds if the worst-case distribution is of a type it can handle – this is the case for triangle finding on graphs of low maximum degree as we will see. Ideally, to prove our lower bound, we would like to consider a hard distribution supported *only* on graphs with low max-degree. The low max-degree condition makes finding a cycle harder and hence proving the lower bound easier. To see this, consider the contrapositive in the special case of triangle finding. If there were no degree bound, then an edge queried at step t could help the algorithm find a very large number of *wedges*, i.e., length-2 paths. In the extreme case, $\Omega(t)$ wedges could be found. As each wedge can be completed to a triangle, the more wedges the algorithm finds, the easier triangle finding becomes.

[4] While Triangle is a decision problem, it is essentially equivalent to its search variant via a simple reduction as explained in the preliminaries section. Therefore, we often do not distinguish between the search and decision problems.

Unfortunately, a product distribution on edge lists cannot be entirely supported on those with low max-degree. In other words, while sampling a graph with high max-degree is a rare event, it is always *possible*. The main technical contribution of our work is to develop tools to handle such rare events within the recording query framework. We name these tools the Mirroring and Exclusion lemmas. They can be used in conjunction to isolate and bound the contribution to the algorithm's success probability when rare events occur. These tools extend the recording query framework to treat any non-product distribution that results from conditioning a product distribution on the absence of rare events.

Theorem 6 is also interesting because the lower bound curiously matches the best-known upper bound on the quantum query complexity of 3-distinctness, which is $Q(3\text{-}\mathsf{DIST}_m) \leq O(m^{5/7})$, up to logarithmic factors. This upper bound was first proven by Belovs [11] over a decade ago using a ground-breaking learning graph algorithm [10]. There have been no improvements since. On the other hand, the best-known lower bound on $Q(3\text{-}\mathsf{DIST}_m)$ is *still* $\Omega(m^{2/3})$, which is simply inherited from the element distinctness lower bound of Aaronson and Shi [1] proven over two decades ago. This is despite many attempts at raising the lower bound using the polynomial method [18, 36], adversary method [40, 39], and even recording queries (equivalently, compressed oracle) method [32]. In Table 1, we summarize these results together with those to be discussed later. Note that 3-CYCLE is equivalent to Triangle.

Table 1 Best known quantum query complexity bounds for k-DIST and k-CYCLE on length-m inputs. For k-CYCLE, we assume the input has max-degree $\leq O(\log(m)/\log\log(m))$. For $k = 3$, the k-DIST lower bound is inherited from that for $k = 2$ [1]. Can k-CYCLE help us close the long-standing gaps for k-DIST?

Problem	k-DIST Lower Bound	k-DIST Upper Bound	k-CYCLE Tight Bound
$k = 3$	$\Omega(m^{2/3})$ [1]	$O(m^{5/7})$ [11]	$m^{5/7\pm o(1)}$ (this work)
$k \geq 4$	$\widetilde{\Omega}(m^{3/4-1/(4k)})$ [36]	$O(m^{3/4-1/(2^{k+2}-4)})$ [11]	$m^{3/4-1/(2^{k+2}-4)\pm o(1)}$ (this work)

We strongly believe that the matching of the lower bound in Theorem 6 and the best-known upper bound on $Q(3\text{-}\mathsf{DIST}_m)$ is *not a coincidence*. This is because the "collision structure" of a uniformly random edge list on n vertices containing $m = \Theta(n)$ edges resembles that of a candidate worst-case distribution on inputs to $3\text{-}\mathsf{DIST}_m$. By collision structure of an edge list, we mean the number of wedges and triangles it contains; by collision structure of an input to 3-distinctness, we mean the number of 2-collisions and 3-collisions it contains, where an l-collision refers to a length-l tuple of indices at which the input contains the same symbol. (For more details about the resemblance, see the overview of cycle finding below.) Indeed, the similarities between the collision structures were so striking that we were led to ask whether Belovs's learning graph algorithm for 3-distinctness could be adapted to provide a matching upper bound to Theorem 6. As our second main result, we answer this question affirmatively.

▶ **Theorem 7** (Informal statement of Theorem 5.27 and Corollary 5.28 of the full version)**.** *Given a length-m edge list on $n = \Theta(m)$ vertices with maximum degree at most $O(\log(m)/\log\log(m))$, we construct a quantum algorithm witnessing*

$$Q(\mathsf{Triangle}) \leq m^{5/7+o(1)}. \tag{8}$$

In particular, the upper bound holds if the edge list is uniformly random, which corresponds to a random sparse graph. Moreover, the algorithm is an adaptation of Belovs's learning graph algorithm for 3-distinctness as developed in [11].

Taken together, Theorems 6 and 7 give the *first* example of a problem for which Belovs's learning graph algorithm in [11] is provably optimal, up to an $m^{o(1)}$ factor.[5] Theorem 6 also suggests that any quantum algorithm that polynomially improves the best-known $O(m^{5/7})$ upper bound on $Q(3\text{-}\mathsf{DIST}_m)$ must exploit more than just the collision structure of the problem, which seems improbable to us.

2.4 Cycle Finding

Our results above suggest that triangle finding in a graph of low maximum degree is closely tied to 3-distinctness. Indeed, they share essentially the same quantum query upper bound and have quite similar quantum algorithms. The 3-distinctness problem generalizes to a well-studied family of problems known as k-distinctness, where $k \geq 3$ is an integer (treated as a constant) and the goal is to decide whether an input list contains k distinct positions with the same symbol. Is there a family of problems generalizing triangle finding in the edge list model that can be similarly tied to k-distinctness?

The answer is yes with *k-cycle* finding being the sought for family. To see why, let us return to the concept of collision structure mentioned above. Let $k\text{-}\mathsf{DIST}_m$ denote the function computing k-distinctness on length-m inputs. A candidate worst-case distribution on inputs to $k\text{-}\mathsf{DIST}_m$ is the uniform distribution over inputs containing exactly one k-collision and $\Omega(m)$ many $(k-1)$-collisions, cf. [3, Section 5] and [13, Section 3]. The intuition is that the many $(k-1)$-collisions present many dead-ends that effectively hide the location of the k-collision. In comparison, writing $P(n,k) \coloneqq n!/(n-k)!$, a uniformly random length-m edge list on $n = \Theta(m)$ vertices contains

$$\begin{aligned} &\Theta(P(n,k)\cdot(m/n^2)^{k-1}) = \Theta(m) \text{ length-}(k-1) \text{ paths} \\ \text{and}\quad &\Theta(P(n,k)\cdot(m/n^2)^{k}) \quad = \Theta(1) \text{ length-}k \text{ cycles.} \end{aligned}$$

Since a length-$(k-1)$ path is analogous to a $(k-1)$-collision and a length-k cycle is analogous to a k-collision, the collision structure of the uniformly random edge list resembles that of the aforementioned worst-case distribution on inputs to k-distinctness. This is also a good point to remark that the uniform distribution over inputs to k-distinctness, which is studied in [32], *cannot* be made to resemble the worst-case distribution (for any setting of input length and alphabet size) because it has the property that the number of $(k-1)$-collisions cannot be too far from the number of k-collisions.

Our main result for cycle finding generalizes our results for triangle finding as follows.

▶ **Theorem 8** (Informal statement of Corollaries 6.2 and 6.17 of the full version)**.** *Given a length-m edge list on $n = \Theta(m)$ vertices with maximum degree at most $O(\log(m)/\log\log(m))$, the quantum query complexity of finding a k-cycle is*

$$Q(k\text{-}\mathsf{CYCLE}) = m^{3/4 - 1/(2^{k+2}-4) \pm o(1)}. \tag{9}$$

Furthermore, this holds if the edge list is uniformly random, which corresponds to a random sparse graph.

The upper bound is *exactly* the same as the best known upper bound for k-distinctness up to an $m^{o(1)}$ factor and is again shown by adapting Belovs's learning graph algorithm for k-distinctness in [11]. We have summarized the state of affairs in Table 1.

[5] Note that the best known quantum query lower bound for k-DIST for every $k \geq 4$ (see [18, 36]) is also polynomially far from the upper bound witnessed by Belovs's learning graph algorithm.

The comparison of collision structures above suggests that the hardness of k-cycle and k-distinctness spring from a common source, which hints at the possibility of lifting our nearly tight lower bound for k-cycle to a nearly tight lower bound for k-distinctness. In fact, we propose Conjecture 6.18 in the full version, stating that the quantum query complexity of k-cycle should not decrease too much under a restriction of the input edges, and prove in Proposition 6.22 of the full version that it enables such lifting.

2.5 Subsequent work

After the appearance of this work, Belovs [12] established the tight quantum query lower bound for k-distinctness, using methods inspired by Zhandry's recording query technique and independently of the lifting conjecture discussed above.

3 Technical overview

We now highlight some of our work's main technical contributions. We inherit the notation above.

3.1 TriangleEdge and Hiding

To lower and upper bound $Q(\mathsf{HIDE}_m[\mathrm{ED}_d])$, our main observation is that the problem "self-reduces" to a more structured version of itself, $\mathsf{HIDE}'_m[\mathrm{ED}_d]$, whose query complexity is easier to characterize. The inputs to $\mathsf{HIDE}'_m[\mathrm{ED}_d]$ are restricted to have the form of d blocks each of length m/d,[6] where each block contains $(m/d-1)$ $*$s and exactly one non-$*$ symbol. Remarkably, imposing the block structure causes no decrease in hardness.

Observe that $\mathsf{HIDE}'_m[\mathrm{ED}_d]$ can be viewed as the composition of ED_d with the so-called pSearch function that extracts the unique non-$*$ symbol from m/d symbols. The latter function has query complexity $\Theta(\sqrt{m/d})$ and a composition theorem [17, Theorem 9] yields $Q(\mathsf{HIDE}_m[\mathrm{ED}_d]) \geq Q(\mathsf{HIDE}'_m[\mathrm{ED}_d]) \geq \Omega(\sqrt{m/d} \cdot d^{2/3}) = \Omega(\sqrt{m}d^{1/6})$. To upper bound $Q(\mathsf{HIDE}_m[\mathrm{ED}_d])$, we consider a quantum algorithm that first randomly permutes the positions of a given input string x. A standard probability argument – like that used to bound the maximum load in a balls-into-bins experiment – implies that the resulting string $\tilde{x}$ is highly likely to be in a block form similar to inputs of $\mathsf{HIDE}'_m[\mathrm{ED}_d]$, except each block may contain up to $\log(d)$ non-$*$ symbols. Since the number of non-$*$ symbols in each block is so small, we simply Grover search for all of them on the fly while running the quantum algorithm for $\mathrm{ED}_{d\log(d)}$ on $d\log(d)$ symbols. This algorithm has query complexity $\widetilde{O}(\sqrt{m}d^{1/6})$.

By considering the block structure, we also see $Q(\mathsf{HIDE}_m[f]) = \Omega(\sqrt{m/d}\cdot Q(f))$ for *every* $f\colon \Sigma^d \to \{0,1\}$. We show $Q(\mathsf{HIDE}_m[f]) = \widetilde{O}(\sqrt{m/d}\cdot Q(f))$ for symmetric $f\colon \{0,1\}^d \to \{0,1\}$ using the characterization of the optimal quantum query algorithm for such functions in [8]. We show $R(\mathsf{HIDE}_m[f]) = O((m/d)\cdot R(f))$ for symmetric $f\colon \Sigma^d \to \{0,1\}$, where $R(\cdot)$ denotes the worst-case (bounded-error) randomized query complexity, using the characterization of the optimal randomized query algorithm for such functions in [7].

[6] We may assume $m/d \in \mathbb{Z}$ without loss of generality. We will not make further remarks like this in the technical overview.

3.2 TriangleVertex and Shuffling

To obtain $Q(\mathsf{TriangleVertex}) \leq O(\sqrt{m}d^{1/4})$ in Proposition 4.1 of the full version, we use a quantum walk algorithm that walks on the Hamming graph with vertices labeled by $r \coloneqq \lceil d^{3/4} \rceil$ positions from part A (the part containing edges incident to the target vertex) of the input. Since we do not a-priori know where part A is, we perform amplitude amplification in both the setup and update steps of the quantum walk to keep the walk on part A. Importantly, this uses the fact that the underlying random walk is uniformly random on the Hamming graph. To lower bound $Q(\mathsf{TriangleVertex}) \geq \Omega(\sqrt{m/d} \cdot Q(\text{3-}\mathsf{DIST}))$, we give a reduction from $\mathsf{HIDE}_m[\text{3-}\mathsf{DIST}]$ to $\mathsf{TriangleVertex}$ in Proposition 4.2 of the full version.

We obtain an exponential separation between $Q(f)$ and $Q(\mathsf{SHUFFLE}[f])$ when f is defined by the graph property $\mathcal{P}$ from [16, Section 6] in the adjacency list model as follows. [16] showed that computing f witnesses an exponential separation between randomized and quantum query complexities. But the quantum query complexity of computing $\mathsf{SHUFFLE}[f]$ is polynomially related to its randomized query complexity since the problem is symmetric [20].

If f is defined by a graph property in the adjacency matrix model, the above argument cannot work since [16] showed that the quantum query complexity of computing f is polynomially related to its randomized query complexity. Nonetheless, we found that an unbounded separation between $Q(f)$ and $Q(\mathsf{SHUFFLE}[f])$ can be witnessed by the following Majority-of-Majority function on a restricted partial domain, that we name ΣMAJ:[7]

▶ **Definition 9.**

$$\Sigma\mathrm{MAJ}_n \colon D_0 \dot{\cup} D_1 \subseteq \{0,1\}^{n^2} \to \{0,1\}, \tag{10}$$

where $\Sigma\mathrm{MAJ}_n(x) = 0$ *if and only if* $x \in D_0$ *and*

1. $x = (x_{1,1}, x_{1,2}, \ldots, x_{n,n}) \in \{0,1\}^{n^2}$ *is in* D_0 *if and only if there exists a subset* $S \subseteq [n]$ *of size* $\geq 2n/3$ *such that for all* $i \in S$, $x^i \coloneqq (x_{i,1}, \ldots, x_{i,n})$ *has Hamming weight* $\left|x^i\right| \geq 2n/3$ *and for all* $i \in [n] - S$, $\left|x^i\right| \leq n/3$.
2. $x = (x_{1,1}, x_{1,2}, \ldots, x_{n,n}) \in \{0,1\}^{n^2}$ *is in* D_1 *if and only if there exists a subset* $S \subseteq [n]$ *of size* $\leq n/3$ *such that for all* $i \in S$, $x^i \coloneqq (x_{i,1}, \ldots, x_{i,n})$ *has Hamming weight* $\left|x^i\right| \geq 2n/3$ *and for all* $i \in [n] - S$, $\left|x^i\right| \leq n/3$.

In other words, inputs in D_0 have at least $2n/3$ "dense" rows, i.e., a substring of the form $x_{i,1}x_{i,2}\ldots x_{i,n}$ for some $i \in [n]$, with at least $2n/3$ ones; and at most $n/3$ "sparse" rows with at most $n/3$ ones. In contrast, inputs in D_1 have at most $n/3$ dense rows and at least $2n/3$ sparse rows.

It is not hard to see that $R(\Sigma\mathrm{MAJ}_n) = O(1)$ as follows. For a given row, we can test whether it is dense or sparse using $O(1)$ queries. Since the fractions of dense blocks for inputs in D_0 and D_1 differ by a constant, we can distinguish between these cases using $O(1)$ queries. Therefore $Q(\Sigma\mathrm{MAJ}_n) \leq R(\Sigma\mathrm{MAJ}_n) = O(1)$.

In contrast, computing $\mathsf{SHUFFLE}[\Sigma\mathrm{MAJ}_n]$ seems at least as hard as finding two distinct input symbols $(x_{i_1,j_1}, (i_1, j_1))$ and $(x_{i_2,j_2}, (i_2, j_2))$ that came from the same row pre-shuffling, i.e., $i_1 = i_2$ but $j_1 \neq j_2$, which is a collision-type problem. Formally, we prove $R(\mathsf{SHUFFLE}[\Sigma\mathrm{MAJ}_n]) = \Omega(\sqrt{n})$ in Theorem 4.6 of the full version by showing that two particular distributions, one supported on the set D_0 and another on D_1, are hard to distinguish by any few-query randomized algorithm using a hands-on total variation distance argument. Therefore, $Q(\mathsf{SHUFFLE}[\Sigma\mathrm{MAJ}_n]) = \Omega(R(\mathsf{SHUFFLE}[\Sigma\mathrm{MAJ}_n])^{1/3}) = \Omega(n^{1/6})$, where the first equality uses [20], which applies since $\mathsf{SHUFFLE}[\Sigma\mathrm{MAJ}_n]$ is symmetric.

[7] We chose this name because Σ resembles a rotated M and $\Sigma\mathrm{MAJ}_n$ is (a restriction of) the composition of two MAJ_ns.

3.3 Triangle Finding (lower bound)

We prove our lower bound on triangle finding (Theorem 6) in Zhandry's recording query framework [43]. Following the framework, we define a "progress quantity" that tracks the progress the algorithm has made in "recording" the searched-for object in its internal memory. The progress quantity can be roughly thought of as the square root of the probability with which the quantum algorithm can find the searched-for object, where the probability is over randomness in *both* the input distribution and the algorithm. The progress quantity depends on the number of queries the quantum algorithm makes. If this quantity is small after the last query, then the algorithm cannot find what it is searching for with high probability.

Our proof has two steps:[8]

1. we first show that the progress in recording much more than $r^*(t) := t^{3/2}\log^2(n)/\sqrt{n}$ wedges in t queries is negligible;
2. then we show that, given we record $O(r^*(t))$ wedges in t queries, the progress of recording a triangle at the $(t+1)$-th query increases by at most $O(\sqrt{r^*(t)}/n)$, which corresponds to the square root of the probability that a random edge completes one of the $r^*(t)$ recorded wedges to a triangle.

Therefore, at the T-th query, the progress of recording a triangle is $\sum_{t=0}^{T}\sqrt{r^*(t)}/n$, which equals $O(T^{7/4}\log(n)/n^{5/4})$, and is $o(1)$ unless $T \geq \Omega(n^{5/7}/\log^{4/7}(n))$. The ability to perform this type of step-by-step analysis is a known strength of the recording query framework.[9] For example, it was exploited to great effect by Liu and Zhandry [32] in proving their tight lower bound on the quantum query complexity of *average-case* k-distinctness.

What is new to our work is how we perform step (i) above. As previously discussed, the issue is that a newly queried edge could contribute to more than one wedge. Let us now see how this issue manifests itself at a technical level. We begin by following the recording queries framework and define a progress quantity $\Lambda_{t,r} \in [0,1]$ for integer t, r with $t \geq 0$ where $\Lambda_{t,r}^2$ represents the probability a quantum query algorithm has recorded at least r wedges immediately after the t-th query. Directly using existing techniques in the framework gives the following recurrence for $\Lambda_{t,r}$:

$$\Lambda_{t,r} \leq \Lambda_{t-1,r} + O(\sqrt{t/n}) \cdot \Lambda_{t-1,r-t+1}, \tag{11}$$

where the factor $O(\sqrt{t/n}) = O(\sqrt{tn/n^2})$ arises as the square root of the probability that a randomly chosen edge is incident to one of the at most $t-1$ edges that can be recorded after the $(t-1)$-th query; the subscript $r-t+1 = r-(t-1)$ arises from the possibility of the new edge recorded at the t-th query contributing $t-1$ additional wedges. However, solving Equation (11) leads to a trivial lower bound for triangle finding that does not even beat the $\Omega(m^{2/3})$ lower bound it inherits from element distinctness.

The main problem with Equation (11) is the subscript $r-t+1$ on the second term on the right-hand side. However, the event it corresponds to seems unlikely to happen when the input is a sparse graph and t is large: if the new edge contributes $t-1$ additional wedges, it must be incident to a degree-$\Omega(t)$ vertex recorded by the quantum query algorithm. Now, our input is a random sparse graph whose maximum degree is at most $O(\log(n)/\log\log(n)) \leq O(\log(n))$ with high probability, *independent of* t. Does this property also hold for the internal memory

[8] The arguments here are better understood by considering n, which represents the number of vertices in the graph. But recall that Theorem 6 concerns the regime $m = \Theta(n)$, so all results here can also be expressed in terms of m.

[9] To quantum query lower bound experts: the standard quantum adversary method [2, 24] is *not* well-suited to performing this type of step-by-step analysis because it gives only weak lower bounds for small success probabilities, and step (ii) needs the progress in step (i) to be inverse-polynomially small to work. If we were forced to redo this analysis using the adversary method, we would have to switch to its *multiplicative* version, see, e.g., [5, 31, 27].

of the quantum query algorithm doing the recording? Our first technical contribution, the Mirroring Lemma (Lemma 5.18 of the full version), answers this question affirmatively. Informally, it says that if a structural event is rare under the initial input distribution, then the corresponding event remains rare in the recording register of any quantum query algorithm, up to a controllable leakage term coming from the change of basis between the standard and recording-query pictures.

▶ **Lemma 10** (Informal statement of the Mirroring Lemma)**.** *Consider running a quantum query algorithm on an input sampled from a product distribution. Suppose that at each position of the input we designate certain symbols as "special". Then the probability that the algorithm records many special symbols after* any *number of queries is at most the sum of:*

1. *the probability that the original sampled input contains many special symbols; and*
2. *a leakage term measuring how much the recording-query change of basis can turn non-special symbols in the original input into special symbols in the recording register.*

Moreover, when every input position contains a special symbol with probability p, the leakage term decays as p^d, where d is the excess of special symbols in the recording register over the original input.

For each vertex v, we apply the Mirroring Lemma by taking the special symbols to be those that are incident to v. Since a sparse graph specified by a uniformly random edge list contains a vertex of degree $\Omega(\log(n))$ with negligible probability, the lemma implies that any quantum algorithm also has negligible probability of recording a vertex with degree $\Omega(\log^2(n))$. Thus, even if the algorithm makes t queries with t large, the number of new wedges created by one additional recorded edge can still be bounded by $O(\log^2(n))$, which is much better than the trivial bound of t. Directly using this technique allows us to improve Equation (11) to

$$\Lambda_{t,r} \leq \Lambda_{t-1,r} + O(\sqrt{t/n}) \cdot \Lambda_{t-1,r-O(\log^2(n))} + \epsilon, \tag{12}$$

where $\epsilon > 0$ is a small number corresponding to the tail probability of the input graph having a vertex of degree $\Omega(\log(n))$.

Unfortunately, Equation (12) still does not yield the desired result: to see this, note that the solution to a similar recurrence $A_{t,r} = A_{t-1,r} + pA_{t-1,r-1} + \epsilon$ (with $p, \epsilon \in [0,1]$ and boundary conditions $A_{0,0} = 1$ and $A_{0,r} = 0$ for all $r > 0$) is $A_{t,r} = \binom{t}{r}p^r + \epsilon(1 + (1+p) + \cdots + (1+p)^{t-1})$. Even for an exponentially small ϵ, the term $\epsilon(1+p)^{t-1}$ blows up for large t. Our second technical contribution, the Exclusion Lemma (Lemma 5.14 of the full version), allows us to overcome this problem. To employ this lemma, we introduce a new progress quantity called $\Lambda'_{t,r}$ that is defined like $\Lambda_{t,r}$ except we additionally require the quantum query algorithm to *not* have recorded a degree-$\Omega(\log(n))$ vertex at any point before the t-th query. By definition, $\Lambda'_{t,r}$ satisfies recurrence Equation (12) with ϵ set to 0, that is,

$$\Lambda'_{t,r} \leq \Lambda'_{t-1,r} + O(\sqrt{t/n}) \cdot \Lambda'_{t-1,r-O(\log^2(n))}. \tag{13}$$

The Exclusion Lemma allows us to upper bound $\Lambda_{t,r}$ by $\Lambda'_{t,r} + O(t\epsilon)$, where the second term no longer blows up for large t and is easy to make negligible. Therefore, solving Equation (13) first for $\Lambda'_{t,r}$ and then using $\Lambda_{t,r} \leq \Lambda'_{t,r} + O(t\epsilon)$ yields the claimed result of step (i).

3.4 Triangle Finding (upper bound)

We prove our upper bound on triangle finding (Theorem 6) by adapting Belovs's learning graph algorithm for 3-distinctness from [11]. Formally, a "learning graph algorithm" is a directed acyclic graph that encapsulates a solution to a semi-definite program that characterizes quantum query complexity [37, 38]. Our main adaptation of Belovs's algorithm pertains to its handling of so-called *faults* in [11, Section 6].

The notion of a fault is easier to explain in Jeffery and Zur's interpretation of Belovs's algorithm as a quantum walk [26].[10] The following explanation is based on [26, Section 1.3]. The quantum walk first creates a uniform superposition over subsets R_1 of indices of some size r_1, and queries all r_1 indices. Then, for each R_1 in the superposition, the algorithm creates a uniform superposition over all subsets R_2 (disjoint from R_1) of indices of some size r_2. But rather than querying every index in R_2, the algorithm *only* queries those $i_2 \in R_2$ that have a *match* in R_1, i.e., $x_{i_2} = x_{i_1}$ for some $i_1 \in R_1$, where x is the input. This significantly reduces query complexity by exploiting the structure of 3-distinctness: any two unequal symbols could not be part of a 1-certificate. Unfortunately, it also leads to the aforementioned faults. The issue is that when performing the update step of the quantum walk by adding a new index j_1 to R_1, we cannot afford the queries needed to update a corresponding R_2 by searching for a $j_2 \in R_2$ such that $x_{j_2} = x_{j_1}$ because R_2 was not fully queried. But if we do not search and there does exist $j_2 \in R_2$ with $x_{j_2} = x_{j_1}$, then the set of queried indices in R_2 becomes incorrect, introducing a fault.

In our setting, there can be more faults because "matching" in our case naturally needs to be redefined to mean: $i_2 \in R_2$ matches with $i_1 \in R_1$ if and only if x_{i_2} (which is an edge in our case) is *incident* to x_{i_1}. In particular, i_2 could match i_1 even if $x_{i_2} \neq x_{i_1}$. However, the number of faults introduced is bounded above by using the maximum degree d of the input, which for a random sparse graph satisfies $d \leq O(\log(n)/\log\log(n))$. Then we adapt the "error-correcting" technique of [11] to correct $O(d)$ faults by paying a multiplicative factor of $2^{O(d)} = n^{o(1)}$ on the quantum query complexity, which leads to the theorem. Along the way, we construct a learning graph algorithm that may be easier to understand than that in [11]; for example, our algorithm genuinely corresponds to a graph, unlike that in [11].

3.5 Cycle Finding

We prove our lower and upper bounds on k-cycle finding by generalizing our proofs in the $k = 3$ (triangle) case.[11] Here is a sketch of how this works.

To prove the lower bound, we consider a uniformly random length-m edge list on n vertices such that $n = \Theta(m)$. Our proof proceeds in $(k-1)$ steps. We use the Mirroring Lemma in the first step for the same reason it was used in the triangle case. On the other hand, the role of the Exclusion Lemma is expanded "by a factor of order k" as it serves to "glue" together the $(k-1)$ steps. For $i \in \{1, \ldots, k-2\}$, we show in the i-th step that it is hard for a t-query algorithm to record much more than $r_{i+1}(t)$ length-$(i+1)$ paths without having recorded at least $r_j(t)$ length-j paths for some $1 \leq j \leq i$. Here, the $r_l(t)$s are certain positive integers essentially *defined* by the relation $r_l(t) = t\sqrt{r_{l-1}(t)/n}$ and boundary condition $r_1(t) = t + 1$. The term $\sqrt{r_{l-1}(t)/n}$ corresponds to the square root of the probability that a single uniformly random edge e extends one of the $r_{l-1}(t)$ length-$(l-1)$ paths to a length-l path. (Note that the precise expression of this probability is $r_{l-1}(t) \cdot 2(n-l)/(n(n-1)/2)$, which is $\Theta(r_{l-1}(t)/n)$.) One technical challenge here (that was not present in the triangle case) is the need to account for the scenario of edge e creating a length-l path by joining together a length-a path with a length-b path such that a, b are non-negative integers and $a+b+1 = l$ *but* neither a nor b equals 0. However, we find that this scenario can be neglected

[10] [26] goes much beyond merely interpreting Belovs's algorithm. However, in this paper, we will only use [26] to aid our explanation of Belovs's algorithm.

[11] While these generalized proofs fully generalize those in the case $k = 3$, we encourage the interested reader to read the proofs in the $k = 3$ case first as they contain most of the key ideas and should make the generalized proofs easier to follow.

as its effect is dominated by that of the extension scenario, i.e., the scenario where a or b is 0. After concluding the first $(k-2)$ steps, we show in the $(k-1)$-th step that it is hard for a t-query algorithm to record a k-cycle without having recorded at least $r_{k-1}(t)$ length-$(k-1)$ paths. Finally, we use the Exclusion Lemma $(k-1)$ times to glue together the results of all steps and show that it is hard for a few-query algorithm to record a k-cycle.

On the upper bound side, our algorithm for k-cycle remains an adaptation of Belovs's algorithm for k-distinctness. The adaptation again pertains to the handling of faults due to a natural redefinition of what matching means. By the degree bound, the maximum number of faults that can occur is fewer than $2d$ where d is the maximum degree of the input edge list. We again adapt Belovs's error correcting technique to handle them by paying a multiplicative factor of $2^{O(dk)}$, which is $m^{o(1)}$ when $d \leq O(\log(m)/\log\log(m))$.

Interestingly, the exact same $r_l(t)$s mentioned in the lower bound overview play a "dual" role in both Belovs's learning graph algorithm for k-distinctness and our adaptation of it to k-cycle: given t queries, the learning graph can be thought of as "recording" $r_j(t)$ number of j-collisions (in the case of k-distinctness) or length-j paths (in the case of k-cycle) at the j-th step, or "stage" as Belovs calls it. Therefore, our lower bound can be interpreted as saying that the learning graph algorithm for k-cycle is tight at every stage.

References

1 Scott Aaronson and Yaoyun Shi. Quantum lower bounds for the collision and the element distinctness problems. *Journal of the ACM*, 51(4):595–605, 2004. doi:10.1145/1008731.1008735.

2 Andris Ambainis. Quantum lower bounds by quantum arguments. In *Proceedings of the 32nd ACM Symposium on Theory of Computing (STOC)*, pages 636–643, 2000. doi:10.1145/335305.335394.

3 Andris Ambainis. Quantum walk algorithm for element distinctness. *SIAM Journal on Computing*, 37(1):210–239, 2007. doi:10.1137/S0097539705447311.

4 Andris Ambainis, Andrew M. Childs, François Le Gall, and Seiichiro Tani. The quantum query complexity of certification. *Quantum Info. Comput.*, 10(3):181–189, 2010. doi:10.26421/QIC10.3-4-1.

5 Andris Ambainis, Robert Špalek, and Ronald de Wolf. A new quantum lower bound method,: with applications to direct product theorems and time-space tradeoffs. In *Proceedings of the 38th ACM Symposium on Theory of Computing (STOC)*, pages 618–633, 2006. doi:10.1145/1132516.1132604.

6 Simon Apers, Frédéric Magniez, Sayantan Sen, and Dániel Szabó. Quantum Property Testing in Sparse Directed Graphs. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2025)*, volume 353, pages 32:1–32:24, 2025. doi:10.4230/LIPIcs.APPROX/RANDOM.2025.32.

7 Ziv Bar-Yossef, Ravi Kumar, and D. Sivakumar. Sampling algorithms: lower bounds and applications. In *Proceedings of the 33rd ACM Symposium on Theory of Computing (STOC)*, pages 266–275, 2001. doi:10.1145/380752.380810.

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

9 Paul Beame, Niels Kornerup, and Michael Whitmeyer. Quantum time-space tradeoffs for matrix problems. In *Proceedings of the 56th ACM Symposium on Theory of Computing (STOC)*, pages 596–607, 2024. doi:10.1145/3618260.3649700.

10 Aleksandrs Belovs. Span programs for functions with constant-sized 1-certificates. In *Proceedings of the 44th ACM Symposium on Theory of Computing (STOC)*, pages 77–84, 2012. doi:10.1145/2213977.2213985.

11 Aleksandrs Belovs. Learning-graph-based quantum algorithm for k-distinctness. In *Proceedings of the 53rd IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 207–216, 2012. doi:10.1109/FOCS.2012.18.

12 Aleksandrs Belovs. Tight quantum lower bound for k-distinctness, 2026. arXiv:2604.05133.

13 Aleksandrs Belovs, Andrew M. Childs, Stacey Jeffery, Robin Kothari, and Frédéric Magniez. Time-efficient quantum walks for 3-distinctness. In *Proceedings of the 40th International Colloquium on Automata, Languages, and Programming (ICALP)*, pages 105–122, 2013. doi:10.1007/978-3-642-39206-1_10.

14 Aleksandrs Belovs and Ansis Rosmanis. On the power of non-adaptive learning graphs. *computational complexity*, 23(2):323–354, 2014. doi:10.1007/s00037-014-0084-1.

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

16 Shalev Ben-David, Andrew M. Childs, András Gilyén, William Kretschmer, Supartha Podder, and Daochen Wang. Symmetries, graph properties, and quantum speedups. In *Proceedings of the 61st IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 649–660, 2020. doi:10.1109/FOCS46700.2020.00066.

17 Gilles Brassard, Peter Høyer, Kassem Kalach, Marc Kaplan, Sophie Laplante, and Louis Salvail. Key establishment à la Merkle in a quantum world. *Journal of Cryptology*, 32(3):601–634, 2019. doi:10.1007/s00145-019-09317-z.

18 Mark Bun, Robin Kothari, and Justin Thaler. The polynomial method strikes back: tight quantum query bounds via dual polynomials. In *Proceedings of the 50th ACM Symposium on Theory of Computing (STOC)*, pages 297–310, 2018. doi:10.1145/3188745.3188784.

19 Joseph Carolan. Compressed permutation oracles, 2025. doi:10.48550/arXiv.2509.18586.

20 André Chailloux. A note on the quantum query complexity of permutation symmetric functions. In *Proceedings of the 10th Innovations in Theoretical Computer Science Conference (ITCS)*, volume 124, pages 19:1–19:7, 2019. doi:10.4230/LIPIcs.ITCS.2019.19.

21 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 *Advances in Cryptology – EUROCRYPT 2021*, pages 598–629, 2021. doi:10.1007/978-3-030-77886-6_21.

22 Amin Shiraz Gilani, Daochen Wang, Pei Wu, and Xingyu Zhou. Hiding, shuffling, and cycle finding: Quantum algorithms on edge lists, 2026. arXiv:2412.17786.

23 Yassine Hamoudi and Frédéric Magniez. Quantum time–space tradeoff for finding multiple collision pairs. *ACM Transactions on Computation Theory*, 15(1–2):1–22, 2023. doi:10.1145/3589986.

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

25 Stacey Jeffery, Robin Kothari, and Frédéric Magniez. Nested quantum walks with quantum data structures. In *Proceedings of the 24th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1474–1485, 2013. doi:10.1137/1.9781611973105.106.

26 Stacey Jeffery and Sebastian Zur. Multidimensional quantum walks. In *Proceedings of the 55th ACM Symposium on Theory of Computing (STOC)*, pages 1125–1130, 2023. doi:10.1145/3564246.3585158.

27 Stacey Jeffery and Sebastian Zur. The compressed oracle is a worthy (multiplicative) adversary, 2025. arXiv:2509.07876.

28 J. Kallaugher. A quantum advantage for a natural streaming problem. In *Proceedings of the 62nd IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 897–908, 2022. doi:10.1109/FOCS52979.2021.00091.

29 François Le Gall. Improved quantum algorithm for triangle finding via combinatorial arguments. In *Proceedings of the 55th IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 216–225, 2014. doi:10.1109/FOCS.2014.31.

30 Troy Lee, Frédéric Magniez, and Miklos Santha. Improved quantum query algorithms for triangle finding and associativity testing. In *Proceedings of the 24th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1486–1502, 2013. doi:10.1007/s00453-015-0084-9.

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

32 Qipeng Liu and Mark Zhandry. On finding quantum multi-collisions. In *Advances in Cryptology – EUROCRYPT 2019*, pages 189–218, 2019. doi:10.1007/978-3-030-17659-4_7.

33 Fermi Ma and Hsin-Yuan Huang. How to construct random unitaries. In *Proceedings of the 57th ACM Symposium on Theory of Computing (STOC)*, pages 806–809, 2025. doi:10.1145/3717823.3718254.

34 Frédéric Magniez, Miklos Santha, and Mario Szegedy. Quantum algorithms for the triangle problem. *SIAM Journal on Computing*, 37(2):413–424, 2007. doi:10.1137/050643684.

35 Christian Majenz, Giulio Malavolta, and Michael Walter. Permutation superposition oracles for quantum query lower bounds. In *Proceedings of the 57th ACM Symposium on Theory of Computing (STOC)*, pages 1508–1519, 2025. doi:10.1145/3717823.3718266.

36 Nikhil S. Mande, Justin Thaler, and Shuchen Zhu. Improved approximate degree bounds for k-distinctness. In *Proceedings of the 15th Conference on the Theory of Quantum Computation, Communication and Cryptography (TQC)*, volume 158, pages 2:1–2:22, 2020. doi:10.4230/LIPIcs.TQC.2020.2.

37 Ben W. Reichardt. Span programs and quantum query complexity: The general adversary bound is nearly tight for every Boolean function. In *Proceedings of the 50th IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 544–551, 2009. doi:10.1109/FOCS.2009.55.

38 Ben W. Reichardt. Reflections for quantum query algorithms. In *Proceedings of the 22nd ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 560–569, 2011. doi:10.5555/2133036.2133080.

39 Ansis Rosmanis. Adversary lower bound for element distinctness with small range, 2014. arXiv:1401.3826.

40 Robert Spalek. Adversary lower bound for the orthogonal array problem, 2013. arXiv:1304.0845.

41 David Wagner. A generalized birthday problem. In *Advances in Cryptology — CRYPTO 2002: 22nd Annual International Cryptology Conference*, pages 288–304, 2002. doi:10.1007/3-540-45708-9_19.

42 Zhiying Yu and Shalev Ben-David. Quantum algorithms for hypergraph simplex finding, 2024. arXiv:2409.00239.

43 Mark Zhandry. How to record quantum queries, and applications to quantum indifferentiability. In *Advances in Cryptology – CRYPTO 2019: 39th Annual International Cryptology Conference*, pages 239–268, 2019. doi:10.1007/978-3-030-26951-7_9.