A Tight Quantum Algorithm for Multiple Collision Search

Xavier Bonnetain[1], Johanna Loyer[2], André Schrottenloher[3], and Yixin Shen[3]

[1] Université de Lorraine, CNRS, Inria, Nancy, France
[2] Inria Saclay
[3] Univ Rennes, Inria, CNRS, IRISA, Rennes, France

Abstract. Searching for collisions in random functions is a fundamental computational problem, with many applications in symmetric and asymmetric cryptanalysis. When one searches for a *single* collision, the known quantum algorithms match the query lower bound. This is not the case for the problem of finding *multiple* collisions, despite its regular appearance as a sub-component in sieving-type algorithms.
At EUROCRYPT 2019, Liu and Zhandry gave a query lower bound $\Omega(2^{m/3+2k/3})$ for finding 2^k collisions in a random function with m-bit output. At EUROCRYPT 2023, Bonnetain et al. gave a quantum algorithm matching this bound for a large range of m and k, but not all admissible values. Like many previous collision-finding algorithms, theirs is based on the MNRS quantum walk framework, but it *chains* the walks by reusing the state after outputting a collision.
In this paper, we give a new algorithm that tackles the remaining non-optimal range, closing the problem. Our algorithm is tight (up to a polynomial factor) in queries, and also in time under a quantum RAM assumption. The idea is to extend the chained walk to a regime in which several collisions are returned at each step, and the "walks" themselves only perform a single diffusion layer.

Keywords: Quantum algorithms, quantum walks, MNRS framework, multiple collision search, quantum cryptanalysis.

1 Introduction

The collision search problem asks to find pairs of (distinct) inputs of a function that have the same output. It is a fundamental problem in quantum algorithms and cryptography, arguably only second to unstructured search. Many applications require to find many such collisions, which we can formulate as follows.

Problem 1 (Multiple collision search). Given access to a (random) function $f : \{0,1\}^n \to \{0,1\}^m$, where $n \leq m \leq 2n$, given $k \leq 2n - m$, find 2^k collisions of f, i.e., pairs of distinct x, y such that $f(x) = f(y)$.

The constraints on the parameters n, m, k ensure that such collisions exist and that we are not asking for more collisions than the expected number of them ($\Theta(2^{2n-m})$ for a random function f of this form).

Tightness of Existing Algorithms. Classically, it is well known that one can find 2^k collisions in $\mathcal{O}(2^{(m+k)/2})$ time and queries to f (which is also a lower bound), against $\mathcal{O}(2^{m/2})$ for a single collision. Indeed, a list of $2^{(m+k)/2}$ random queries to f can be expected to contain (the order of) $2^{(m+k)-m} = 2^k$ collisions. This simple algorithm is tight.

In the quantum setting, Aaronson and Shi [1] showed a query lower bound $\Omega(2^{m/3})$ for a single collision, which is matched by the BHT [6] and Ambainis' [2] algorithms for different values of m, thereby solving the problem for $2^k = 1$.

For a general k, Liu and Zhandry [14] showed a lower bound $\Omega(2^{2k/3+m/3})$. It is common knowledge that by extending the BHT algorithm to multiple outputs, one can reach the bound for $k \leq 3n-2m$, in queries to f, and also in time if one assumes quantum-accessible classical memory. The *chained quantum walk* introduced in [3] reaches the same complexity for $k \leq \min(2n-m, m/4)$, in queries, and in time if one assumes quantum-accessible quantum memory. Unfortunately, the union of these two parameter ranges does not entirely cover all n, k, m such that $n \leq m \leq 2n$ and $k \leq 2n-m$. A range of (k, m), forming a triangle contained in $[n/3; n] \times [n; 1.6n]$, remains only covered by basic generalizations of both algorithms which reach a suboptimal query and time complexity.

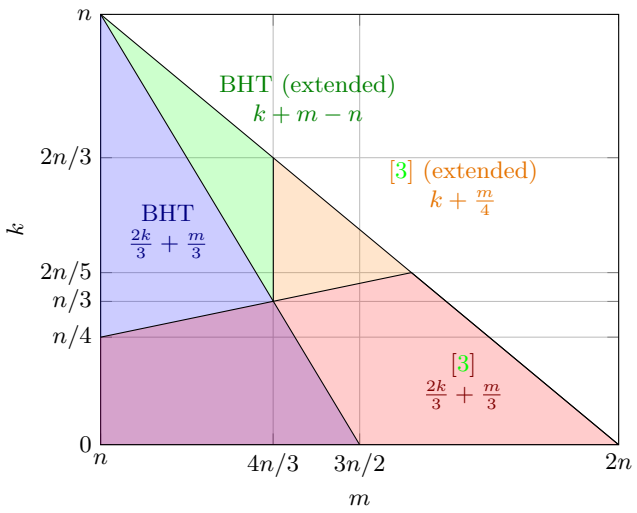


Fig. 1. Parameter ranges and complexity exponents of quantum multiple collision search algorithms. This figure is taken from [3]. (All complexities are both in queries, and gate count, assuming qRAM gates).

Contribution. In this paper, we give a new algorithm that covers this remaining range. Like in [3], we use a *chained* version of the MNRS quantum walk framework [15]. The idea of the chained quantum walk is to reuse the final state of

a walk as starting state for a new one. Indeed, the state of a walk is a superposition of subsets of $\{0,1\}^n$. At the end of a walk, one expects the subsets to always contain a collision: one can *extract* and measure this collision, collapsing the vertex on a superposition of subsets which is still close to uniform – and sufficient to carry on.

In this new version, we increase the vertex size so much that in fact, the quantum state always contains (exponentially many) collisions on average. Now, there is no need to properly *walk* during the quantum walks, and they are reduced to a single diffusion layer. The effect of the diffusion is to re-randomize the vertex, making new collisions appear.

This allows us to prove the following:

Theorem 1. *For all $n \leq m \leq 2n$, $k \leq 2n-m$, there exists a quantum algorithm solving the multiple collision search problem in $\widetilde{\mathcal{O}}\left(2^{2k/3+m/3}\right)$ queries, memory and total gate count (assuming qRAM gates). Furthermore, for any $\ell \leq 2k/3 + m/3$, there exists a quantum algorithm using memory 2^ℓ and $\widetilde{\mathcal{O}}\left(2^{k+m/2-\ell/2}\right)$ queries and total gate count (assuming qRAM gates).*

With the lower bound from [14] this implies, as a direct corollary:

Corollary 1 (Complexity of Multiple collision search). *The query complexity of Problem 1 is, up to a factor polynomial in n, $2^{m/3+2k/3}$.*

Applications in Symmetric Cryptography. In symmetric cryptography, the *limited birthday* problem asks, given black-box oracle access to a permutation E of $\{0,1\}^n$ (a block cipher), to find many pairs P, P' such that $E(P) \oplus E(P')$ and $P \oplus P'$ both belong to a given vector subspace of $\{0,1\}^n$. It is an important part of impossible differential attacks [4], and also serves as a distinguisher of random permutations [12]. The limited birthday problem is solved by restricting the inputs of E (or outputs) to *structures* and solving a multiple collision search problem on such structures. Depending on the sizes of the restricting vector subspaces, one can encounter both cases with *few* collisions to output, or all of them, making this a very versatile application [8].

Applications in Asymmetric Cryptography. Sieving algorithms are currently the fastest at solving the shortest vector problem on integer lattices, and are used to estimate the security of post-quantum lattice-based cryptosystems. They also exist in the cryptanalysis of code-based schemes [9,10]. Sieving algorithms construct several lists of elements (e.g., vectors) where the next ones are obtained by finding partial collisions between the previous ones. Previously, the generic improvement of multiple collision-finding in [3] allowed to improve (slightly) the previous best quantum algorithm for lattice sieving given in [7]. In this paper, we do not improve lattice sieving (since we only improve multiple-collision search in an adjacent regime), but we believe that our new technique could likely be used inside similar algorithms.

Finding multicollisions. The problem of multicollision-finding in the quantum setting has only been studied in the case $n = m$ [14]. As the known algorithms in this case start by computing a large number of collisions, a tight algorithm to find many collisions is likely a first step towards a tight quantum algorithm for multicollision-finding in the case $m > n$.

2 Preliminaries

We assume knowledge of the basics of quantum computing in the circuit model [16] such as qubits, quantum states, unitaries, *etc.* In order to solve Problem 1 in the quantum setting, the function f is accessed as a unitary oracle:

$$|x\rangle |y\rangle \xmapsto{O_f} |x\rangle |y \oplus f(x)\rangle \quad ,$$

where x is an n-bit input and y an m-bit input.

We estimate the complexities of our algorithms asymptotically in n, m and k, as follows. The *query complexity* is the number of queries to O_f (which also includes classical queries to f as a special case). The *time complexity* (or gate count) is the number of elementary quantum gates; the *memory complexity* is the number of qubits or classical bits of memory. Since we are only interested in asymptotic complexities, any set of universal quantum gates is sufficient; one can take the Clifford+T gate set.

In order to reach optimal time complexities, we need *quantum RAM* (qRAM). Different variants of qRAM are used in collision search algorithms: for example, the BHT algorithm requires only *quantum-accessible classical memory* [6]. In our case, we need *quantum-accessible quantum memory*. Like previous works [2], we formalize our qRAM assumption by introducing the so-called qRAM gate:

$$|y_1, \ldots, y_M\rangle |x\rangle |i\rangle \xmapsto{\mathsf{qRAM}} |y_1, \ldots, y_{i-1}, x, y_{i+1}, \ldots, y_M\rangle |y_i\rangle |i\rangle \quad .$$

We give cost 1 to the qRAM gate (like other elementary gates) even if M is exponential-size; the qRAM gate can even span the entire quantum circuit. While this is obviously a very powerful model, it has no incidence on the query complexity, since a qRAM gate can always be simulated by an exponential-size Clifford+T circuit.

2.1 Quantum Search

Let $|\psi_U\rangle$ be a quantum state (the "uniform superposition") produced by some **Setup** algorithm, which is a superposition of two orthogonal components $|\psi_G\rangle$ (the "good subspace") and $|\psi_B\rangle$ (the "bad subspace"):

$$|\psi_U\rangle = \beta |\psi_B\rangle + \alpha |\psi_G\rangle$$

for some parameters α, β. Let $Ref_{|\psi_B\rangle}$ be the reflection over $|\psi_B\rangle$, that performs:

$$Ref_{|\psi_B\rangle} |\psi_B\rangle = |\psi_B\rangle , \quad Ref_{|\psi_B\rangle} |\psi_G\rangle = - |\psi_G\rangle \quad .$$

We assume that this reflection can be performed by an oracle O_f; that is, there is a Boolean function f defined on the outputs of **Setup** that returns 1 iff an output belongs to the good subspace. We can either consider a *phase oracle* for f that immediately implements $Ref_{|\psi_B\rangle}$, or a computational oracle.

Grover's algorithm [11] and more generally amplitude amplification [5], consist in alternating reflections over $|\psi_B\rangle$, which are implemented using a *checking oracle* that recognizes the components of $|\psi_G\rangle$, and reflections over $|\psi_U\rangle$, which can be implemented by calling the **Setup** algorithm. Indeed, it can be shown that $Ref_{|\psi_U\rangle}Ref_{|\psi_B\rangle}$ is a rotation of angle $2\arcsin\alpha$ in the plane spanned by $|\psi_B\rangle, |\psi_G\rangle$.

We view amplitude amplification as a way to transform a linear combination of $|\psi_B\rangle, |\psi_G\rangle$ into another.

Lemma 1. *Assume that $|\psi_U\rangle = \beta|\psi_B\rangle + \alpha|\psi_G\rangle$. Suppose that $\alpha \leq \frac{1}{\sqrt{2}}$. Starting from $|\psi_U\rangle$ or $|\psi_B\rangle$, there exists $k = \mathcal{O}(1/\alpha)$ such that, after k iterations of $Ref_{|\psi_U\rangle}Ref_{|\psi_B\rangle}$, the resulting state is a linear combination $\beta'|\psi_B\rangle + \alpha'|\psi_G\rangle$ where $\alpha' = \mathcal{O}(1)$.*

Proof. We have:

$$|\psi_U\rangle = \sin\theta|\psi_G\rangle + \cos\theta|\psi_B\rangle$$

where $\theta = \arcsin\alpha$, so after k iterations we obtain:

$$\cos((\arcsin\alpha) + 2k(\arcsin\alpha))|\psi_B\rangle + \sin((\arcsin\alpha) + 2k(\arcsin\alpha))|\psi_G\rangle \quad .$$

We have:

$$\begin{aligned}\sin((\arcsin\alpha) + 2k(\arcsin\alpha)) &\geq (2k+1)\arcsin\alpha - ((2k+1)\arcsin\alpha)^2/6 \\ &\geq (2k+1)\arcsin\alpha - ((2k+1)\alpha\pi)^2/24 \quad .\end{aligned}$$

This can be made a constant by choosing $k = \mathcal{O}(1/\alpha)$. If we start from $|\psi_B\rangle$, the initial angle is 0 instead, but the asymptotic behavior is the same. □

One can be more precise with the number of iterations required to achieve α' close to 1, but this is not necessary for us since we focus on asymptotic complexities only.

Lemma 2. *Suppose that $\alpha \leq \frac{1}{\sqrt{2}}$. Starting from $|\psi_G\rangle$, there exists $k = \mathcal{O}(1/\alpha)$ such that after k iterations of $Ref_{|\psi_U\rangle}Ref_{|\psi_B\rangle}$, the resulting state is a linear combination $\beta'|\psi_B\rangle + \alpha'|\psi_G\rangle$ where $\beta' = \mathcal{O}(1)$.*

Proof. The initial angle in the plane spanned by $|\psi_B\rangle, |\psi_G\rangle$ is $\frac{\pi}{2}$ in that case, so after k iterations we have:

$$\begin{aligned}&\cos(\pi/2 + 2k(\arcsin\alpha))|\psi_B\rangle + \sin(\pi/2 + 2k(\arcsin\alpha))|\psi_G\rangle \\ &\qquad = \sin(2k(\arcsin\alpha))|\psi_B\rangle - \cos(2k(\arcsin\alpha))|\psi_G\rangle \quad .\end{aligned}$$

Thus, by choosing $k = \mathcal{O}(1/\alpha)$ we get a constant amplitude on $|\psi_B\rangle$. □

In both cases, we navigate in the plane spanned by $|\psi_G\rangle, |\psi_B\rangle$. If we assume that one of them (their roles can be exchanged) represents a "typical" case and the other an "atypical" case (with an amplitude α in the uniform superposition), then Lemma 1 shows that we can move from the typical case to the atypical one with $\mathcal{O}(1/\alpha)$ iterations. Conversely, Lemma 2 shows that one can move from the atypical case back to the typical one with $\mathcal{O}(1/\alpha)$ iterations.

After performing the iterations, we will project the result and see if we obtain $|\psi_B\rangle$ or $|\psi_G\rangle$. If we did not obtain the wanted state, we can just restart. The average number of repetitions will be constant. The result is summarized in Corollary 2, and the procedures are given in Algorithm 1, which allows to transform $|\psi_B\rangle$ into $|\psi_G\rangle$, *and the converse*, at the same cost.

Corollary 2. *Assume that* $|\psi_U\rangle = \beta|\psi_B\rangle + \alpha|\psi_G\rangle$ *where* α *is at most a constant. There are two quantum procedures using an expected* $\mathcal{O}(1/\alpha)$ *calls to* $(Ref_{|\psi_U\rangle}Ref_{|\psi_B\rangle})$ *that respectively:*

- *On input* $|\psi_B\rangle$ *or* $|\psi_U\rangle$*, return* $|\psi_G\rangle$*;*
- *On input* $|\psi_G\rangle$*, return* $|\psi_B\rangle$*.*

Algorithm 1 Procedure Search$(|\psi_B\rangle, |\psi_G\rangle, |\psi_U\rangle)$.

Define: $|\psi_U\rangle = \beta|\psi_B\rangle + \alpha|\psi_G\rangle$ where $|\psi_B\rangle$ and $|\psi_G\rangle$ are orthogonal; an oracle f to recognize states in $|\psi_G\rangle$

Input: $|\psi\rangle \leftarrow |\psi_B\rangle$ or $|\psi_U\rangle$ (resp. $|\psi_G\rangle$)
Output: $|\psi\rangle \leftarrow |\psi_G\rangle$ (resp. $|\psi_B\rangle$)

Success $\leftarrow$ False
while Not Success **do**
  $|\psi\rangle \leftarrow (Ref_{|\psi_U\rangle}Ref_{|\psi_B\rangle})^{1/\alpha}|\psi\rangle$
  Compute f: $|\psi\rangle \leftarrow |\psi\rangle|f(\psi)\rangle$
  Measure the last bit (projects on $|\psi_B\rangle$ or $|\psi_G\rangle$)
  If we obtained the wanted state, Success $\leftarrow$ True
end while
Return $|\psi\rangle$

2.2 MNRS Quantum Walks

The MNRS quantum walk framework [15] can be seen as an extension of quantum amplitude amplification where the implementation of $Ref_{|\psi_U\rangle}$ differs from the **Setup** algorithm. We recall here some essential features, but for a more comprehensive account, one can refer to [3].

One considers a regular undirected graph $G = (V, E)$ with degree d, vertex set V and edge set E, where a proportion ε of the vertices are *marked* (noted $M \subseteq V$). The goal of the quantum walk is to recover a marked vertex. Assume

that an encoding of vertices as orthogonal quantum states is given, denoted as $|\hat{x}\rangle$ for $x \in V$. For $x \in V$, let N_x be the set of its neighbors. The MNRS quantum walk is analogous to a random walk on *edges* of the graph G. One defines:

$$|p_x\rangle := \frac{1}{\sqrt{d}} \sum_{y \in N_x} |\hat{y}\rangle$$

$$|\psi_U\rangle := \frac{1}{\sqrt{V}} \sum_{x \in V} |\hat{x}\rangle |p_x\rangle, \qquad |\psi_M\rangle := \frac{1}{\sqrt{M}} \sum_{x \in M} |\hat{x}\rangle |p_x\rangle$$

$$A := \mathrm{span}(|\hat{x}\rangle |p_x\rangle)_{x \in V}, \qquad B := \mathrm{span}(|p_x\rangle |\hat{x}\rangle)_{x \in V}$$

The MNRS framework emulates $Ref_{|\psi_U\rangle}$ by a *phase estimation* of the unitary $Ref_B Ref_A$. This phase estimation requires $\mathcal{O}(1/\sqrt{\delta})$ calls to both Ref_B and Ref_A, where δ is the *spectral gap* of the graph G. The spectral gap is related to classical random walks: $\mathcal{O}(1/\delta)$ is the number of random walk steps to take, starting from any vertex, before the current vertex becomes uniformly distributed.

Both Ref_B and Ref_A can be implemented from a unitary that, on input $|\hat{x}\rangle$, produces the superposition of neighbors $|p_x\rangle$. This is what is commonly known as the **Update** unitary. Finally, one needs a **Check** unitary that flips the phase of bad vertices. The algorithm is often constructed so that **Check** has a trivial implementation. By applying Algorithm 1 on the output of **Setup**, we obtain the following.

Theorem 2 (MNRS (informal)). *There is a quantum algorithm that, using 1 call to* ***Setup****, and on expectation* $\mathcal{O}\left(\frac{1}{\sqrt{\varepsilon}}\frac{1}{\sqrt{\delta}}\right)$ *calls to* ***Update*** *and* $\mathcal{O}\left(\frac{1}{\sqrt{\varepsilon}}\right)$ *calls to* ***Check****, returns* $|\psi_M\rangle$*.*

While this point of view suffices for query complexity, the state of a vertex is a very large one. Thus, keeping two entire vertices in memory is wasteful, and having to reconstruct an entire vertex during the **Update** is too costly.

Johnson Graph and Radix Trees. Most MNRS quantum walks in cryptanalysis use *Johnson graphs*, for which this problem can be solved generically: the vertex-vertex pair can be compressed to a single vertex and a *coin* indicating the next one. This is detailed in [3].

A Johnson graph $J(X, R)$ is the graph where the vertices are the subsets of X containing R elements. Two sets are adjacent if they have exactly one differing element. In collision search and similar algorithms, the vertex x is encoded into $|\hat{x}\rangle$ by using a specific set data structure, and possibly computing additional information. In our case we store pairs $(z, f(z))$ as $(f(z)|z)$ using a concatenation of bit-strings.

The quantum radix trees [13] allow both to represent efficiently an ordered set and to perform complex operations efficiently (assuming qRAM), such as adding an element, removing an element and finding an element. We can also augment the radix trees with useful information such as the number of collisions

that each branch contains (in order to quickly find collisions, without having to write them down). All of this side information is part of the encoding $|\hat{x}\rangle$, and also makes trivial any implementation of **Check** that would rely on the number of collisions in a vertex.

2.3 Statistics on the Number of Collisions

In the remainder of this paper, we will work with *multicollision tuples* rather than collision pairs. By multicollision, we mean a unique tuple of elements which all have the same image by f (for example, a 3-collision is not counted as a pair of collisions). Our algorithm will return such tuples.

In a random function on n bits, the largest size of a multicollision is expected to be $\mathcal{O}(n)$. This means that up to a polynomial factor, the number of multicollisions and the number of collisions returned by our algorithm is the same; which is why we can focus on the former. This is more practical and equivalent to Problem 1 up to a polynomial factor. The exceptional case (a function having a larger multicollision than expected by this bound) is taken as a case of failure of our algorithm.

Lemma 3 ([17, Theorem 2]). *Let be n, m, ℓ with $n \leq m$. Then the probability $P(n, m, \ell)$ that a random function $f : X \to Y$ where $|X| = 2^n$ and $|Y| = 2^m$ has a ℓ-multicollision satisfies $P(n, m, \ell) \leq 2^{-m(\ell-1)} \cdot \binom{2^n}{\ell}$.*

Corollary 3. *There exist constants ε and c such that with probability $1 - \varepsilon$, all multicollisions in a random function are tuples of size $\leq cn$.*

Proof. Let $\ell = cn$. By applying the previous lemma and the standard bound $\binom{2^n}{\ell} \leq (e2^n/\ell)^\ell$, we get $P(n, m, \ell) \leq \left(2^{-n(1-1/\ell)} \frac{e2^n}{\ell}\right)^{cn}$ as $n \leq m$. For large n, the union bound over all $\ell \geq cn$ shows that a multicollision of size larger than cn exists with negligible probability $\varepsilon \leq 2^{-\Omega(n)}$. □

Let us denote $X = \{0,1\}^n$ and $Y = \{0,1\}^m$. As our main algorithm considers mutiple walks, we will need to define reduced sets X' and Y' and a corresponding function $f' : X' \to Y'$ which is simply a restriction of f.

We are interested in the number of multicollisions that can appear in a vertex. The vertex is a set S' of elements taken uniformly at random from X', with $|S'| < |X'|$. All three of $|S'|$, $|X'|$ and $|Y'|$ can be varying parameters, but we will ensure that they do not vary too much throughout the algorithm, so that our statistics remain valid from one walk to another.

We use the following heuristic.

Heuristic 1. The number of multicollisions in a set follows a Poisson distribution. When its expectation is large, the number of multicollisions follows a normal distribution.

Under this heuristic, we obtain the following result, which is our main tool to handle the statistics of collisions in vertices.

Lemma 4. *Let $Z(S')$ be the number of multicollisions (of f') in the set S', with elements selected uniformly at random from X'. Assume that $R/2 \leq |S'| \leq R$ and $M/2 \leq |Y'| \leq M$ for some global parameters R and M, where $8R < M$. There is a constant c such that, for any pair of constants c_1, c_2, the probability:*

$$\Pr_{S'}\left[c\frac{|S'|^2}{|Y'|} - c_1\frac{R}{\sqrt{M}} \leq Z(S') \leq c\frac{|S'|^2}{|Y'|} + c_2\frac{R}{\sqrt{M}}\right] \tag{1}$$

is also constant.

Proof. Let us write $Z := Z(S')$ for simplicity. By definition, we have $Z = \#\{y \in Y' : N_y \geq 2\}$ with $N_y := \#\{x \in S' : f'(x) = y\}$. By Heuristic 1, we can approximate $p := \Pr(N_y \geq 2) = 1 - e^{-\lambda} - \lambda e^{-\lambda}$ with mean $\lambda = |S'|/|Y'|$. For $\lambda \in (0, 1/4)$, we have $\lambda^2/2 \leq 1 - e^{-\lambda}(1+\lambda) \leq 2\lambda^2/3$. Note that $\lambda < 1/4$ follows from $8R < M$, and so $4|S'| < |Y'|$ for any values of S' and Y'. We have $\mathbb{E}[Z] = |Y'| \cdot p$, hence

$$\frac{|S'|^2}{2|Y'|} \leq \mathbb{E}[Z] \leq \frac{2|S'|^2}{3|Y'|}. \tag{2}$$

Now we aim to bound the variance. For distinct $y, z \in Y'$, the events $(N_y \geq 2)$ and $(N_z \geq 2)$ are negatively correlated, since having more elements $x \in S'$ with images $f'(x) = y$ decreases the chance that another $z \in Y'$ admits at least two preimages in S'. So the covariance of these two events is negative, and thus

$$\mathrm{Var}(Z) = \sum_y \mathrm{Var}(\mathbb{1}_{(N_y \geq 2)}) + 2\sum_{y<z} \mathrm{Cov}(\mathbb{1}_{(N_y \geq 2)}, \mathbb{1}_{(N_z \geq 2)}) \leq \sum_y \mathrm{Var}(\mathbb{1}_{(N_y \geq 2)})$$

We have $\mathrm{Var}(\mathbb{1}_{(N_y \geq 2)}) = p(1-p) \leq p$, then $\mathrm{Var}(Z) \leq \mathbb{E}[Z]$. By Equation (2), the standard deviation then satisfies $\sigma_Z \leq \sqrt{2/3}\frac{|S'|}{\sqrt{|Y'|}} \leq C\frac{R}{\sqrt{M}}$ for constant $C > 0$, as $|S'| \leq R$ and $|Y'| \geq M/2$.

For any fixed constant $c > 0$, we deduce

$$\Pr\left(\frac{|S'|^2}{2|Y'|} - cC\frac{R}{\sqrt{M}} \leq Z \leq \frac{2|S'|^2}{3|Y'|} + cC\frac{R}{\sqrt{M}}\right) \leq \Pr(|Z - \mathbb{E}[Z]| \leq c\sigma_Z)$$

and by applying Chebyshev's inequality, we know this probability is at least $1 - 1/c^2$, which is constant. □

When we take a random subset of size R with codomain of size M, with many multicollisions, the average number of multicollisions is $E = \mathcal{O}(R^2/M)$, and the standard deviation is $T = \mathcal{O}(R/\sqrt{M})$. As a consequence of Lemma 4 we can say that a constant proportion of subsets have a number of multicollisions in the interval $[E; E+T]$; and also, a constant proportion fall in the interval $[E-T; E]$. Note that the technical condition $8R < M$ in Lemma 4 is also obviously validated in our applications, as the vertex size is at most of order $\mathcal{O}(2^n)$, and the domain size at least $\mathcal{O}(2^n)$, so we can always leave a constant factor of difference between these two.

We can even go further. Because of the evolution of R and M throughout our algorithm, the average of the number of multicollisions will vary as well. But if we don't change R and M "too much", then we still have constant probability to fall into the intervals with the new value of E'.

Lemma 5. *Fix $T = \mathcal{O}\left(R/\sqrt{M}\right)$ and assume $R \ll M$. Let $R' \in [R-T; R]$ and $M' \in [M-T; M]$. Let $E' = c\frac{(R')^2}{M'}$ where c is the universal constant of Lemma 4. Then the number of collisions in a random vertex of size R falls in the intervals $[E'; E'+T]$ and $[E'-T; E']$ with constant probability.*

Proof. We simply prove that the difference between E' and E is smaller than the standard deviation. Indeed, we have:

$$E' = \frac{(R')^2}{M'} = \frac{(R-x)^2}{M-y} \tag{3}$$

for some $x, y \in [0; R/\sqrt{M}]$, which is much smaller than both R and M. Thus:

$$\begin{aligned} E' &= \frac{R^2}{M}\left(1-\frac{x}{R}\right)^2\left(1-\frac{y}{M}\right)^{-1} \\ &\simeq \frac{R^2}{M}\left(1-\frac{2x}{R}+\frac{y}{M}\right) \quad . \end{aligned}$$

We notice then that: $\frac{R^2}{M}\frac{x}{R} = \frac{Rx}{M} \leq \frac{R^2}{M^{3/2}} \ll \frac{R}{\sqrt{M}}$ as we have $R \ll M$. Besides: $\frac{R^2}{M}\frac{y}{M} \ll y = R/\sqrt{M}$. □

In summary, if the parameters of the set are modified up to an amount equal to the standard deviation of the number of collisions; then our reasoning on intervals still holds.

2.4 Main Ideas

In order to explain our main idea, we first briefly recall the algorithms of [2] and [3] (with slight modifications).

Ambainis' element distinctness algorithm [2] can be reframed as an MNRS quantum walk on a Johnson graph $J(\{0,1\}^n, R)$, where vertices are subsets of $\{0,1\}^n$ of size R. A vertex is marked if it contains a collision. Therefore, the probability for a vertex to be marked is $\mathcal{O}\left(R^2/2^m\right)$. The spectral gap is $\delta = 1/R$. The cost of **Setup** is $\mathcal{O}(R)$, the cost of **Update** is negligible (using a quantum radix tree), and the total gate count of the algorithm is: $\widetilde{\mathcal{O}}\left(R + \frac{2^{m/2}}{R}\sqrt{R}\right)$ which is optimized by setting $R = 2^{m/3}$ and obtaining $\widetilde{\mathcal{O}}\left(2^{m/3}\right)$.

Chained Walk. In the *chained* quantum walk, the goal is to obtain many collisions. One starts with a vertex of size R, and performs a first quantum walk. At the end of this walk, one obtains the uniform superposition of marked vertices, i.e., subsets of $\{0,1\}^n$ containing at least one (multi)collision. These (multi)collisions are *removed* from the subset, and measured. Crucially, the rest of the vertex is not measured. The remaining elements form a uniform superposition over:

- Subsets of size $R - t$ where t is the number of elements measured;
- Excluding the elements of $\{0,1\}^n$ which were measured;
- Excluding all other elements of $\{0,1\}^n$ which would have an image equal to the images measured;
- Which *do not* contain a (multi)collision.

This is the *extraction* step of the walk. After outputting this collision, the remaining state can be reused as the starting state in a new walk, with a restricted function f and a reduced vertex size. Note that this starting state is actually a superposition of *unmarked* vertices (that do not contain a multicollision), but this is enough for quantum search.

The gate count is of order:

$$R + 2^k \frac{2^{m/2}}{R} \sqrt{R} \ . \tag{4}$$

One can use any vertex size R as long as $R \leq 2^{m/2}$: afterwards, vertices contain on average one collision or more, and this formula is not valid anymore.

Therefore, in order to reach the optimal complexity $\widetilde{\mathcal{O}}(2^{\frac{m}{3}+\frac{2k}{3}})$ (obtained for $R = 2^{\frac{m}{3}+\frac{2k}{3}}$), one needs to have $k \leq \frac{m}{4}$. In other words, the algorithm fails when *the number of collisions to output is too high* with respect to m, and is not optimal in a regime with intermediate m (e.g, $m = 1.5n$) where we would want to output all collisions.

New Regime. Our new algorithm follows the principle of the chained quantum walk (the final state of a walk is reused as the initial state of the next one), but fits in the regime where a vertex contains more than one (multi)collision on average.

As a simplified view, let us neglect the fact that extracting multicollisions post-selects the remaining elements, modifying their distribution (and the domain of the quantum walk). Indeed, as long as the number of extracted elements remains at most a constant proportion of the entire domain, these modifications do not significantly impact the algorithm, although we will handle them carefully.

Following Lemma 4, a vertex of size R contains $\mathcal{O}(R^2/2^m)$ multicollisions on average, and the standard deviation is $\mathcal{O}(R/2^{m/2})$. This means that if we extract $\mathcal{O}(R/2^{m/2})$ multicollisions and measure them, we collapse on a superposition of vertices which are still *typical*: they only contain slightly less multicollisions than average. Using Corollary 2, we will show that we can always change a *typical* superposition of vertices into another typical one after $\mathcal{O}(1)$ steps of quantum search, and in our case, of quantum walk. We can thus transform our superposition of vertices into vertices having slightly *more* multicollisions than average, repeat the extraction, and so on. Therefore, at the cost of a total time $\widetilde{\mathcal{O}}(\max(\sqrt{R}, R/2^{m/2}))$, we can extract $\mathcal{O}(R/2^{m/2})$ multicollisions. As we have $\sqrt{R} < 2^{m/2} \implies R/2^{m/2} < \sqrt{R}$, the time is dominated by $\widetilde{\mathcal{O}}(\sqrt{R})$.

We find that 2^k is always larger than $(R/2^{m/2})$ and the total complexity is:

$$R + \frac{2^k}{(R/2^{m/2})} \times \sqrt{R} = R + \frac{2^k 2^{m/2}}{\sqrt{R}} \ .$$

Optimizing R, we recover the time $\widetilde{\mathcal{O}}\big(2^{2k/3+m/3}\big)$.

This new regime, together with the previous one of [3] (where the vertex contains less than one multicollision on average) covers all possible parameters for k and m. It also completes the time-memory trade-off curve, where for any $\ell \leq 2k/3 + m/3$, there exists an algorithm with memory $\widetilde{\mathcal{O}}\big(2^\ell\big)$ and time $\widetilde{\mathcal{O}}\big(2^{k+m/2-\ell/2}\big)$: for small values of ℓ the algorithm is the one of [3], and for large values of ℓ it is ours.

3 New Algorithm

This section is dedicated to the details of our new algorithm. We will rely mostly on:

- Corollary 2 to *flip* a superposition of vertices into its complementary, using a constant number of walk steps, as long as the proportion of such vertices is constant;
- Corollary 2 to *correct* a superposition of *atypical vertices* back into a superposition of typical ones;
- Lemma 4 for the statistics on the number of collisions.

3.1 Preliminaries: Restricted Functions

Throughout this section we write $X = \{0,1\}^n$ and $Y = \{0,1\}^m$.

The multicollision tuples returned by our algorithm are stored in a classical, quantum-accessible table C with entries of the form: $u : \{x_1, \ldots, x_r\}$ where $f(x_1) = \ldots = f(x_r) = u$. The set of *images* of C is defined as: $I_C = \{u \in Y, \exists t \subseteq X \text{ s.t. } (u : t) \in C\}$. We have $I_C \subseteq Y$. The set of *preimages* of C is defined as: $P_C = \{x \in X, \exists u \in I_C, f(x) = u\} \subseteq X$. We caution the reader here that P_C is in general bigger than the multicollision tuples stored in C, because C does not necessarily contain all preimages.

We can ensure that throughout the algorithm, $|P_C| < 2^{n-1}$. Indeed, even in the corner case where $n = m$ and we want $\mathcal{O}(2^n)$ collisions, we can stop the algorithm after returning $2^n/\mathcal{O}(n)$ multicollision tuples, and repeat at most $\mathcal{O}(n)$ times afterwards (ultimately, a polynomial factor in the time complexity).

Given a table C, we define a restricted function f_C that excludes all preimages of C, and all corresponding images:

$$\begin{cases} f_C \ : X \backslash P_C \mapsto Y \backslash I_C \\ f_C(x) = f(x) \ . \end{cases} \tag{5}$$

Vertices and Data Structure. Given the table C, given a size parameter R (which varies during our algorithm), we define a set of vertices for one of our walks:

$$V^{R,C} := \{S \subseteq (X \backslash P_C), |S| = R\} \tag{6}$$

and furthermore, we define $V^{R,C}_{x,y} \subseteq V^{R,C}$ as the subset of vertices *containing between x and y multicollisions*. We authorize $y = \infty$ in this notation, where $V^{R,C}_{x,\infty}$ means all vertices containing more than x multicollisions. By definition of P_C, the corresponding images of these multicollisions are ensured to not appear in I_C.

Like in previous work [3], subsets of X are identified with (orthogonal) quantum states thanks to the quantum radix tree data structure [13], and we can operate efficiently on them. Therefore, we can introduce the notation $\left|V^{R,C}_{x,y}\right\rangle := \sum_{S \in V^{R,C}_{x,y}} |S\rangle$ as a uniform superposition of such sets. The set data structure also encodes additional data:

- The outputs of f;
- The number of multicollisions and their locations. This can be added to the radix tree by storing at each node the number of multicollisions that occur in its sub-tree.

Polynomial-time algorithms exist for:

- Inserting an element into the set;
- Checking if an element belongs to the set;
- Creating a uniform superposition of (multi)collisions in the vertex.

3.2 Superposition of Elements

During our algorithm, we populate the table C of multicollisions. However we cannot compute P_C, the set of all their preimages (we only know the preimages which belong to the table C). Despite this limitation, we can still create a uniform superposition over the set $X \backslash P_C$: intuitively, we just have to perform rejection sampling.

Lemma 6. *Given C stored in quantum-accessible memory such that $|C| \leq 2^{n-1}$, there exists a unitary to construct a uniform superposition over $X \backslash P_C$:*

$$|0\rangle \mapsto \sum_{x \in (X \backslash P_C)} |x\rangle \quad . \tag{7}$$

with negligible error and polynomial time.

Proof. The idea is to perform a (quantum) rejection sampling, starting from:

$$\sum_{x \in X} |x\rangle$$

and testing whether $x \in P_C$ efficiently thanks to the quantum RAM access. We obtain:

$$\sqrt{\frac{|X| - |P_C|}{|X|}} \sum_{x \in (X \setminus P_C)} |x\rangle |0\rangle + \sqrt{\frac{|P_C|}{|X|}} \sum_{x \in P_C} |x\rangle |1\rangle \quad .$$

We amplify the part 0 using robust amplitude amplification techniques. In particular, while we do not know the size of P_C, an upper bound of $|P_C| < |X|/2$ is sufficient to succeed in polynomial time. □

By using the same algorithm with an additional set of forbidden elements, we obtain a polynomial-time algorithm that creates a uniform superposition of neighboring vertices, which is sufficient to implement the **Update** unitary of our walks.

3.3 Algorithm

We consider quantum walks on Johnson graphs. The set of vertices for each walk is given by $V^{R,C}$ as defined above, for C the current table of multicollisions, and R the current vertex size. As explained above, R remains similar to the initial vertex size.

Our algorithm (Algorithm 2) alternates between *walk steps* and *extraction steps*, which are detailed later in this section. An extraction step starts from a superposition of typical vertices with more collisions than average, $\left|V^{R,C}_{E,E+T}\right\rangle$, and produces a superposition $\left|V^{R,C}_{E-T,E}\right\rangle$ with less collisions. A walk step starts from $\left|V^{R,C}_{E-T,E}\right\rangle$ and restores a superposition of vertices $\left|V^{R,C}_{E,E+T}\right\rangle$ suitable for the next extraction step.

We start by analyzing the walk steps. By Lemma 4 and Lemma 5, we know that the number of multicollisions in vertices does not vary much; its average depends on R, but only slightly decreases at each loop iteration. We can adapt the number of extracted multicollisions to correct the interval appropriately (we will see later that multicollisions are extracted one by one).

After obtaining the state $\left|V^{R',C'}_{E'-T,E'}\right\rangle$, we transform it into $\left|V^{R',C'}_{E',E'+T}\right\rangle$ using a series of applications of Corollary 2. For each application, we define a different partition of "good" and "bad" vertices depending on the number of multicollisions that they contain. We have defined four intervals which form a partition of all vertices for the walk. As long as we do not obtain the wanted interval $[E'; E'+T]$, applying Corollary 2 and projecting transforms the current interval into another one of the three, where each of them has a constant probability to appear. Consequently this process terminates with an expected constant number of steps. Each step contains an expected constant number of single-step quantum walks.

Algorithm 2 Multiple collision-finding algorithm.

Choose an initial vertex size 2^ℓ

1: Initialize $R \leftarrow 2^\ell$
2: Initialize $C \leftarrow \emptyset$
3: Initialize the state $\left|V^{R,C}\right\rangle$ (uniform superposition of vertices)
4: Project on $\left|V^{R,C}_{E,E+T}\right\rangle$ using Corollary 2

5: **for** $2^k 2^{m/2}/R$ iterations **do**
Extraction step:
6: **Extract** T multicollision tuples and measure them
7: The state becomes: $\left|V^{R',C'}_{E-T,E}\right\rangle$ for a new R', C'
8: **Extract** a little more to correct the state into $\left|V^{R',C'}_{E'-T,E'}\right\rangle$
▷ Here E' is the new expectation value, which depends on R', and is slightly smaller than E
9: Update R and C

Walk step. From now on we write only the intervals on the number of collisions. We start with the internal $[E'-T;E']$ and we want to obtain $[E';E'+T]$.
10: Measure the interval on the number of collisions
▷ Possibilities: $[0;E'-T']$, $[E'-T';E']$; $[E';E'+T']$, $[E'+T';\infty]$, each having a constant probability
11: **Case** $[0;E'-T']$: transform the state into $[E'-T';\infty]$, go to Step 10
12: **Case** $[E'-T;E']$: transform the state into $[0;E'-T'] \cup [E';+\infty]$, go to Step 10
13: **Case** $[E';E'+T]$: go to Step 5
14: **Case** $[E'+T;\infty]$: transform the state into $[0;E'+T']$, go to Step 10
15: **end for**

3.4 Extraction Steps

We detail the *extraction* routine which transforms $\left|V_{E,E+T}^{R,C}\right\rangle$ into $\left|V_{E-T,E}^{R',C'}\right\rangle$ for modified parameters R', C', and returns T multicollision tuples (classical values). Actually, we give a more general routine to extract a single multicollision tuple from a state $\left|V_{x,y}^{R,C}\right\rangle$, where x, y are input parameters.

Basic Idea. Given a vertex V, we know thanks to the tree data structure how many multicollisions appear and where they are located. The basic idea of the extraction routine is to create a uniform superposition of these collisions, outside of the tree, into a separate register, and to measure:

- the size of the register, which projects on multicollisions having the same number of elements (and determines the new parameter R');
- the actual elements of the multicollisions, which gives the new C'.

This transforms the state $\left|V_{x,y}^{R,C}\right\rangle$ into a superposition $\left|V_{x',y'}^{R',C'}\right\rangle$ of vertices which exclude any preimage of the newly measured multicollision, and contain between $x-1$ and $y-1$ collisions.

An Issue with post-Selection. There is a subtlety due to the fact that the vertices in $\left|V_{x,y}^{R,C}\right\rangle$ do not contain the same number of multicollisions. Consider the following example of a superposition with a vertex having 3 collisions $|c, c', c''\rangle$ and a vertex having two collisions $|c, c'\rangle$. Let us simplify the writing of these vertices by considering only the collisions, furthermore we write $|\{c, c'\}\rangle$ for an unordered *set* of elements. We will select one of the collisions and move it to a fixed position in the quantum memory, which we measure. Before measurement, the quantum state has the form:

$$\frac{1}{\sqrt{3}}|c\rangle|\{c', c''\}\rangle + \frac{1}{\sqrt{3}}|c'\rangle|\{c, c''\}\rangle + \frac{1}{\sqrt{3}}|c''\rangle|\{c, c'\}\rangle + \frac{1}{\sqrt{2}}|c\rangle|\{c'\}\rangle + \frac{1}{\sqrt{2}}|c'\rangle|\{c\}\rangle \quad . \tag{8}$$

Now, we measure. Suppose that we obtain c, then the state becomes proportional to the following, non-uniform superposition:

$$\frac{1}{\sqrt{3}}|\{c', c''\}\rangle + \frac{1}{\sqrt{2}}|\{c'\}\rangle \quad . \tag{9}$$

The issue here is that there is a post-selection depending on the number of collisions in the vertices, which is not constant in our algorithm. As we need the superposition to remain uniform, we need to correct this deviation.

Procedure. Let us start from the superposition $\left|V_{x,y}^{R,C}\right\rangle$. In a given vertex V with z multicollisions, let $(t_i)_{1\le i\le z}$ be the multicollision tuples. When we create the uniform superposition of multicollisions, we obtain the state:

$$\sum_{x\le z\le y}\sum_{V\in V_z^{R,C}}|V\rangle\left(\frac{1}{\sqrt{z}}\sum_{1\le i\le z}|t_i\rangle\right) \quad . \tag{10}$$

The problem here is that the amplitude over a given t_i depends on z (the number of multicollisions in the vertex). So we make it constant by adding new "dummy" values d_i:

$$\sum_{x\leq z\leq y}\sum_{V\in V_z^{R,C}}|V\rangle\left(\frac{1}{\sqrt{y}}\sum_{1\leq i\leq z}|t_i\rangle+\frac{1}{\sqrt{y}}\sum_{z+1\leq i\leq y}|d_i\rangle\right)\quad . \tag{11}$$

Now, we measure the last register:

- With probability $\frac{1}{|V_{x,y}^{R,C}|}\sum_{x\leq z\leq y}|V_z^{R,C}|\frac{z}{y}$, we obtain a multicollision. We then collapse on a uniform superposition of vertices $V_{x-1,y-1}^{R',C'}$ where R' is reduced by the size of the multicollision, and C' contains this new multicollision (as a result of post-selection, the vertex excludes any preimage of the new value). This is what we wanted.
- For all $x+1\leq i\leq y$, we measure d_i with probability:

  $$\frac{1}{|V_{x,y}^{R,C}|}\sum_{x\leq z\leq i-1}|V_z^{R,C}|\frac{1}{y}=\frac{1}{y}\frac{|V_{x,i-1}^{R,C}|}{|V_{x,y}^{R,C}|}$$

  In that case we project the vertex on:

  $$\sum_{x\leq z\leq y|z+1\leq i}\sum_{V\in V_z^{R,C}}|V\rangle=\left|V_{x,i-1}^{R,C}\right\rangle\quad . \tag{12}$$

  Indeed, we still have a uniform superposition of sets, and a set belongs in this superposition if and only if it has less than $i-1$ multicollisions (in which case, the dummy $|d_i\rangle$ is created). There are no new multicollisions extracted; we have simply changed the bounds on the number of multicollisions in the superposition of vertices.

The latter case is problematic for us, as we have refined the superposition: this makes the vertex possibly more "atypical". Fortunately, we can always go back to the typical case, at a cost depending on the exact proportion of vertices that we fell upon, using Lemma 2.

Lemma 7. *Assume that $V_{x,y}^{R,C}$, $V_{0,x}^{R,C}$, $V_{y,\infty}^{R,C}$ contain a constant proportion of all vertices $V^{R,C}$, and $x/y=\mathcal{O}(1)$. Then there is a quantum algorithm with* average time $\mathcal{O}\left(\frac{\sqrt{R}(y-x)}{y}\right)$ *that, on input $\left|V_{x,y}^{R,C}\right\rangle$ extracts* on average *a constant number of multicollisions and returns an updated $\left|V_{x-1,y-1}^{R',C'}\right\rangle$.*

Proof. The probability to obtain a multicollision immediately is bigger than x/y, which is constant. If we do not obtain a multicollision, we need to revert back to $\left|V_{x,y}^{R,C}\right\rangle$.

For all $i\geq x$, we project on $\left|V_{x,i}^{R,C}\right\rangle$ with probability $\frac{1}{y}\frac{|V_{x,i}^{R,C}|}{|V_{x,y}^{R,C}|}$. Via a series of a constant number (on expectation) of manipulations, we will transform the interval $[x;i]$ into $[0;x[$, then $[x;y]$.

- Starting from $[x;i]$, we use Corollary 2 to obtain $[0;x] \cup [i;\infty[$ using an expected number of $\mathcal{O}\left(\sqrt{\frac{|V^{R,C}|}{|V^{R,C}_{x,i}|}}\right)$ iterations. With constant probability we project on $[0;x]$ or $[i;\infty[$.
- Starting from $[i;\infty[$, we use Corollary 2 to obtain $[0;i[$ with $\mathcal{O}(1)$ iterations. With constant probability we project on $[0;x]$ or $[x;i]$. (If i is close to x, the probability to project on $[x;i]$ is actually low). In the latter case, we go back to the previous step.

As a consequence, for all $i \geq x$, we succeed later with $\mathcal{O}\left(\sqrt{\frac{|V^{R,C}|}{|V^{R,C}_{x,i}|}}\right)$ iterations on average. By taking the average over all i, the number of iterations that we need is, up to a constant:

$$\sum_{i=x}^{y} \frac{1}{y} \frac{|V^{R,C}_{x,i}|}{|V^{R,C}_{x,y}|} \sqrt{\frac{|V^{R,C}|}{|V^{R,C}_{x,i}|}} = \mathcal{O}\left(\frac{1}{y} \sum_{i=x}^{y} \sqrt{\frac{|V^{R,C}_{x,i}|}{|V^{R,C}_{x,y}|}}\right) = \mathcal{O}\left(\frac{y-x}{y}\right) \quad . \tag{13}$$

Here we use the fact that $|V^{R,C}|$ and $|V^{R,C}_{x,y}|$ are the same up a constant, and then, that $|V^{R,C}_{x,i}| \leq |V^{R,C}_{x,y}|$. Each iteration costs $\sqrt{R}$ queries and $\widetilde{\mathcal{O}}\left(\sqrt{R}\right)$ time, giving the result. □

We use this algorithm for $\left|V^{R,C}_{E,E+T}\right\rangle$ where $x = E$ and $y = E + T$, which satisfy its conditions. The average time is $\mathcal{O}\left(\sqrt{R}T/E\right)$, where $T = \mathcal{O}\left(R/\sqrt{M}\right)$ and $E = \mathcal{O}\left(R^2/M\right)$. Since we need to extract T multicollisions, the total average time is $\mathcal{O}\left(\sqrt{R}T^2/E\right) = \mathcal{O}\left(\sqrt{R}\right)$, the same as the walk step in Algorithm 2.

4 Conclusion

In this paper, we proved that the multiple collision search problem quantum complexity in queries (and time) is $2^{2k/3+m/3}$ where m is the output bit-size and 2^k the number of collisions, closing the remaining gap in some range of k and m. In the new targeted regime, we use MNRS quantum walks as a tool for *diffusion* in the current state rather than *search*, which allows to correct the expected number of collisions in this state – allowing to extract collisions in subsequent steps. This peculiar use of the diffusion operator in the MNRS walk could perhaps have further applications.

References

1. Aaronson, S., Shi, Y.: Quantum lower bounds for the collision and the element distinctness problems. J. ACM 51(4), 595–605 (2004)
2. Ambainis, A.: Quantum Walk Algorithm for Element Distinctness. SIAM J. Comput. 37(1), 210–239 (2007)

3. Bonnetain, X., Chailloux, A., Schrottenloher, A., Shen, Y.: Finding many collisions via reusable quantum walks - application to lattice sieving. In: EUROCRYPT (5). Lecture Notes in Computer Science, vol. 14008, pp. 221–251. Springer (2023)
4. Boura, C., Naya-Plasencia, M., Suder, V.: Scrutinizing and improving impossible differential attacks: Applications to clefia, camellia, lblock and simon. In: ASIACRYPT (1). Lecture Notes in Computer Science, vol. 8873, pp. 179–199. Springer (2014)
5. Brassard, G., Hoyer, P., Mosca, M., Tapp, A.: Quantum amplitude amplification and estimation. Contemporary Mathematics 305, 53–74 (2002)
6. Brassard, G., Høyer, P., Tapp, A.: Quantum cryptanalysis of hash and claw-free functions. In: LATIN. Lecture Notes in Computer Science, vol. 1380, pp. 163–169. Springer (1998)
7. Chailloux, A., Loyer, J.: Lattice sieving via quantum random walks. In: ASIACRYPT (4). Lecture Notes in Computer Science, vol. 13093, pp. 63–91. Springer (2021)
8. David, N., Naya-Plasencia, M., Schrottenloher, A.: Quantum impossible differential attacks: applications to AES and SKINNY. Des. Codes Cryptogr. 92(3), 723–751 (2024), https://doi.org/10.1007/s10623-023-01280-y
9. Ducas, L., Esser, A., Etinski, S., Kirshanova, E.: Asymptotics and improvements of sieving for codes. In: EUROCRYPT (6). Lecture Notes in Computer Science, vol. 14656, pp. 151–180. Springer (2024)
10. Engelberts, L., Etinski, S., Loyer, J.: Quantum sieving for code-based cryptanalysis and its limitations for ISD. Des. Codes Cryptogr. 93(6), 1611–1644 (2025), https://doi.org/10.1007/s10623-024-01545-0
11. Grover, L.K.: A Fast Quantum Mechanical Algorithm for Database Search. In: Proceedings of the Twenty-Eighth Annual ACM Symposium on the Theory of Computing 1996. pp. 212–219. ACM (1996)
12. Hosoyamada, A., Naya-Plasencia, M., Sasaki, Y.: Improved attacks on sliscp permutation and tight bound of limited birthday distinguishers. IACR Trans. Symmetric Cryptol. 2020(4), 147–172 (2020)
13. Jeffery, S.: Frameworks for Quantum Algorithms. Ph.D. thesis, University of Waterloo, Ontario, Canada (2014), http://hdl.handle.net/10012/8710
14. Liu, Q., Zhandry, M.: On finding quantum multi-collisions. In: EUROCRYPT (3). Lecture Notes in Computer Science, vol. 11478, pp. 189–218. Springer (2019)
15. Magniez, F., Nayak, A., Roland, J., Santha, M.: Search via quantum walk. SIAM J. Comput. 40(1), 142–164 (2011)
16. Nielsen, M.A., Chuang, I.: Quantum computation and quantum information (2002)
17. Suzuki, K., Tonien, D., Kurosawa, K., Toyota, K.: Birthday paradox for multicollisions. IEICE Trans. Fundam. Electron. Commun. Comput. Sci. 91-A(1), 39–45 (2008), https://doi.org/10.1093/ietfec/e91-a.1.39