Almost Improving Quantum Lattice Sieving

Mathias Boucher*

University of Rennes

Abstract. The asymptotically fastest known algorithms for the Shortest Vector Problem (SVP) rely on lattice sieving combined with quantum random walks to detect vector collisions. The current best quantum time complexity, due to Bonnetain and al. [Bon+22], is $2^{0.2563d+o(d)}$. Chailloux and Loyer [CL21] showed that an effective strategy for implementing the quantum walk is to solve a SVP instance of smaller dimension during the update phase. In parallel, Heiser [Hei21] proposed an improved quantum sieving algorithm achieving a running time of $2^{0.2571d+o(d)}$, which notably avoids the use of a quantum random walk. In this work, we integrate the algorithm of [Hei21] into the update phase of the quantum walk sieve of [CL21] and analyze its impact on the overall quantum complexity. We obtain a modest improvement for the first algorithm of [CL21], reducing its complexity from $2^{0.2605d+o(d)}$ to $2^{0.2594d+o(d)}$. However, our results indicate that no further asymptotic gains can be achieved through this approach.

1 Introduction

Lattice-based cryptography is a leading candidate for post-quantum security. Its foundations rely on worst-case hardness assumptions for lattice problems, initiated by Ajtai's construction of hard instances of SVP and related tasks [Ajt96]. Lattices also support advanced cryptographic primitives such as fully homomorphic encryption [Gen09], which further motivates understanding their algorithmic hardness. Because the security of modern lattice-based schemes depends on the difficulty of solving the Shortest Vector Problem (SVP), assessing both classical and quantum algorithms for SVP remains essential.

The SVP asks for a shortest non-zero vector in a d-dimensional lattice. It plays a central role in the cryptanalysis of lattice-based constructions, most notably through its use as a subroutine in the BKZ algorithm [BMW19]. Exact SVP can be approached with enumeration algorithms [Kan83], which require little memory but have exponential running time. Otherwize sieving algorithms offer the best known asymptotic complexities at the cost of large memory. The first practical heuristic sieve [NV08] demonstrated that randomized reductions of vector lists can solve SVP at moderate dimensions. Further improvements yielded faster implementations [MV10]. Subsequent work introduced new directions for nearest-neighbor searching on the sphere ([IM98],[Bec+15]), tuple sieving [BLS16], and angular locality-sensitive hashing [Laa15]. These techniques

* Contact: mathias.boucher@irisa.fr

reduce SVP to high-dimensional nearest-neighbor search, which is now a standard viewpoint in sieving algorithms.

Classical sieving achieves a heuristic running time of $2^{0.292d+o(d)}$ using locality-sensitive filtering [Bec+15]. Quantum analogues of these methods exploit quantum search and quantum walks to accelerate the search step. Quantum search on Markov chains [Mag+11] provides a general framework, leading to quantum sieves that outperform their classical counterparts. Early quantum analyses of lattice sieving [Laa15] indicated potential gains. Chailloux and Loyer [CL21] employed quantum random walks equipped with locality-sensitive filtering to reach $2^{0.2570d+o(d)}$. Bonnetain et al. [Bon+22] introduced reusable quantum walks, lowering the exponent to $2^{0.2563d+o(d)}$. Heiser optimized the quantum use of locality-sensitive filtering to obtain a sieve running in time $2^{0.2571d+o(d)}$. By optimizing the method for sampling close filters, he avoids the use of a quantum random walk [Hei21]. Similar improvements are made in the context of tuple sieving [Eng+25]. Understanding how these approaches interact is therefore relevant for assessing the potential of further improvements in quantum sieving.

Contribution. In this work, we integrate Heiser's filter sampling algorithm into the update phase of the quantum walk sieve of Chailloux and Loyer. We analyze the resulting quantum complexity and compare it with existing trade-offs. Our approach yields a slight improvement over the first algorithm of [CL21], reducing its exponent from $2^{0.2605d+o(d)}$ to $2^{0.2594d+o(d)}$. However, our results also indicate that no additional asymptotic speed-up can be achieved by further optimizing this combination.

Technical Contribution

Locality-sensitive filtering (LSF) algorithms preprocess a list of lattice vectors by grouping them into *buckets* of geometrically close vectors, so that one can restrict the search for reducing pairs to vectors that share a common bucket. In the quantum setting, Chailloux and Loyer [CL21] exploit quantum random walks on Johnson graphs to detect such collisions efficiently. In their construction, a vertex v of the Johnson graph $J(L_y, r)$ carries a subset $L_y^v \subset L_y$ of reduced lattice vectors together with a collection of β-buckets $\{J_\beta^v(t)\}_{t\in\mathcal{C}_2}$, where $\mathcal{C}_2$ is a random product code on $\mathbb{S}^{d-2}$. A vertex is marked when two vectors $y_1, y_2 \in J_\beta^v(t)$ form a neighboring pair for some common filter $t \in \mathcal{C}_2$, i.e., they satisfy $|\langle y_1, t\rangle| \geq \cos(\beta)$ and $|\langle y_2, t\rangle| \geq \cos(\beta)$.

Our modification relaxes this marking condition. We instead declare a vertex v marked whenever there exist $y_1, y_2 \in L_y^v$ and $t \in \mathcal{C}_2$ such that $|\langle y_1, t\rangle| \geq \cos(\beta)$ and $|\langle y_2, t\rangle| \geq \cos(\omega)$, where $\omega \geq \beta$. Intuitively, the two vectors no longer need to share the same tight β-bucket: it suffices that a common filter t lies β-close to one vector and ω-close to the other. This asymmetric condition introduces a fourth free parameter c_3, where $N^{c_3} = |L_y^v| \cdot \mathcal{V}_d(\omega)$, alongside the parameters c, c_1, and c_2 of [CL21].

This new marking condition is what enables the integration of Heiser's *filter sampling* technique [Hei21] into the update phase. Filter sampling provides a

subexponential-time procedure that, given a fixed vector $y \in \mathbb{S}^{d-1}$, samples a new filter ω-close to y from the random product code $\mathcal{C}_2$. To exploit this, we increase the size of the vertex data by adding a second auxiliary data structure $D_\omega(v) = \{J^v_\omega(t)\}_{t\in\mathcal{C}_2}$, containing the buckets of vectors that are ω-close (but not β-close) to each filter t. During the update step, instead of performing a Grover search over all β-close filters, one queries $D_\omega(v)$ to locate the relevant filter in time proportional to $\sqrt{|\mathcal{C}_2| \cdot \mathcal{V}_d(\omega) \cdot |L^v_y| \cdot \mathcal{V}_d(\beta)}$, which is sublinear in the original update cost when $\omega > \beta$.

The change of marking condition also affects the ratio ε of marked vertices. We compute a new lower bound on ε, which now depends on $\mathcal{W}_d(\beta, \omega \mid \theta^*_\alpha)$, the probability that a uniformly random filter lies in $\mathcal{H}_{y_1,\beta} \cap \mathcal{H}_{y_2,\omega}$ for a neighboring pair (y_1, y_2) at angle θ^*_α. The setup and checking costs are not affected by these modifications.

Combining these elements yields a quantum time complexity $T^{(c,c_1,c_2,c_3)}_{\mathrm{QRW}}$. The numerical optimization shows that the proposed approach provides a modest improvement for the first algorithm of [CL21], reducing its exponent from $2^{0.2605d+o(d)}$ to $2^{0.2594d+o(d)}$. However, when applied to the faster algorithms of [CL21] and [Bon+22], the optimization yields $c_2 = c_3$, which collapses our construction back to the original marking condition of [CL21]. This is due to the update phase which is already optimal in those algorithms. Filter sampling does not provide additional gain. We therefore conclude that no further asymptotic improvement can be achieved through this approach.

2 Locality Sensitive Filtering Preliminaries

2.1 Sieving Algorithms

Sieving approach. The sieving approach begins by sampling an initial list of lattice vectors. The algorithm then performs a preprocessing step designed to combine these vectors in a systematic way in order to produce a new list of the same size whose elements have smaller norm. Assume that each iteration reduces the norms by a factor γ with $0 < \gamma < 1$. After a polynomial number of iterations, one obtains vectors short enough that the list typically contains a solution to the Shortest Vector Problem.

Nearest neighbor search. The task considered here is the Nearest Neighbor Search (NNS) problem. Let L be a list of vectors sampled from the unit sphere $\mathbb{S}^{d-1}$. The goal is to preprocess L so that one can efficiently identify a pair of vectors that are close to each other. A pair (x, y) in $\mathbb{S}^{d-1}$ is called a *neighboring pair* when

$$|\langle x, y\rangle| > \tfrac{1}{2}.$$

This condition is equivalent to stating that the angle between x and y is strictly less than $\pi/3$, or equivalently that $\|x - y\|^2 < 1$. Locality-sensitive filtering algorithms form a family of methods that use collections of filters, introduced in the next subsection, to solve this problem efficiently.

2.2 Spherical Geometry

α-close pairs. Let x and y two vectors of the sphere $\mathbb{S}^{d-1}$ and $\alpha \in [0, \pi/2]$ be an angle. We say that the pair (x, y) is α-close, or x is α-close to y if

$$|\langle x, y \rangle| \geq \cos(\alpha).$$

We introduce the following notation. If $x, y \in \mathbb{S}^{d-1}$, to say that y and x are β-close, we note :

$$x \overset{\beta}{-} y.$$

This means that the angle between x and y or between $-x$ and y is lower then α. Moreover, if x is β-close to $t \in \mathbb{S}^{d-1}$ and y is ω-close to t, we note it

$$x \overset{\beta}{-} t \overset{\omega}{-} y.$$

One can translate the condition for x, y to be a neighboring pair through this notation by writing $x \overset{\pi/3}{-} y$.

Hypercone. We denote by $\mathcal{H}_{v,\alpha}$ the spherical cap associated to a vector $v \in \mathbb{S}^{d-1}$ and the angle α. It consists in

$$\mathcal{H}_{v,\alpha} := \{\, y \in \mathbb{S}^{d-1} \;;\; y \overset{\alpha}{-} v \,\}.$$

This cap allows us to compute the following probabilities:

Proposition 1 ([Bec+15]). *Let v a fixed element of $\mathbb{S}^{d-1}$. Then the probability $\mathcal{V}_d(\alpha) := \Pr_{y \leftarrow \mathbb{S}^d}(y \overset{\alpha}{-} v)$ does not depend on the choice of v and*

$$\mathcal{V}_d(\alpha) = \mathrm{poly}(d) \cdot \sin^d(\alpha),$$

which is the ratio of the volume of the spherical cap $\mathcal{H}_{\alpha,v}$ to the volume of the sphere $\mathbb{S}^{d-1}$.

Proposition 2 ([Bec+15]). *Let v, w two fixed elements of $\mathbb{S}^{d-1}$ such that their angle is θ. Then the probability $\mathcal{W}_d(\alpha, \beta \mid \theta) := \Pr_{y \leftarrow \mathbb{S}^{d-1}}(v \overset{\alpha}{-} y \overset{\beta}{-} w)$ does not depend on the choice of the pair (v, w) and*

$$\mathcal{W}_d(\alpha, \beta \mid \theta) = \mathrm{poly}(d) \cdot \left(1 - \frac{\cos^2(\alpha) + \cos^2(\beta) - \cos(\alpha)\cos(\beta)\cos(\theta)}{\sin^2(\theta)}\right)^{d/2}$$

which is the ratio of the volume of $\mathcal{H}_{v,\alpha} \cap \mathcal{H}_{w,\beta}$ to the volume of the sphere $\mathbb{S}^{d-1}$.

2.3 Random Product Codes

Introduced by [Bec+15], a Random Product Code (RPC) is a subset of codewords uniformly distributed over $\mathbb{S}^{d-1}$ which admits a fast list decoding. A random product code $\mathcal{C}$ is constructed by making the Cartesian product of m subcodes $C_1, \ldots, C_m$, where each subcode C_i is a set of b vectors uniformly distributed over the sphere $\sqrt{1/m} \cdot \mathbb{S}^{d/m-1}$. In other words, $\mathcal{C} = C_1 \times \cdots \times C_m$. Let L be a list of N lattice vectors. In the context of lattice sieving the parameters m and b used to build $\mathcal{C}$ verify $m = \log(d)$ and $b^m = N^{O(1)}$.

Proposition 3 ([Bec+15]). *Let L be the list of N lattice vectors of dimension d. Let $\mathcal{C} = C_1 \times \cdots \times C_m$ be a random product code of b^m codewords, with $m = \log(d)$ and $b^m = N^{O(1)}$. For any $x \in C$ and $\alpha \in [0, \pi/2]$, one can compute $\mathcal{H}_{x,\alpha} \cap \mathcal{C}$ in time $N^{o(1)}|\mathcal{H}_{x,\alpha} \cap \mathcal{C}|$. Moreover, there exists an algorithm which initializes filters from $\mathcal{C}$ and compute all the buckets in time $N^{1+o(1)}|\mathcal{H}_{x,\alpha} \cap \mathcal{C}|$.*

Remark 1. Because the codewords are uniformly distributed over $\mathbb{S}^{d-1}$, the proposition 1 ensures that the cardinality $|\mathcal{H}_{x,\alpha} \cap \mathcal{C}|$ of the above proposition does not depend on the vector x.

2.4 Locality Sensitive Filtering scheme

Filters and buckets. Let L denote a list of vectors. The goal of a Locality Sensitive Filtering (LSF) scheme is to sample vectors $f_1, \ldots, f_t$ from the unit sphere $\mathbb{S}^{d-1}$; these vectors are called *filters*. For each filter f one associates *α-buckets* $B_\alpha(f)$, where the parameter α is an angle satisfying $\pi/3 < \alpha < \pi/2$. The bucket associated with f is defined by

$$B_\alpha(f) := \mathcal{H}_{\alpha,f} \cap L.$$

LSF scheme steps. The purpose of the scheme is to identify neighboring pairs in L by examining only those vectors that fall into a common bucket, instead of checking all pairs in L. An LSF scheme consists of two steps:

1. Initialization: sample the filters, form the buckets, and insert the vectors of L into the appropriate buckets.
2. Search: inspect the bucket structure to look for neighboring pairs.

Every LSF-based algorithm since [Bec+15] uses random projection codes (RPC) to accelerate the initialization step, which currently appears to be the fastest method for this task. The search step admits two efficient strategies:

- Iterate over each $x \in L$, examine the filters close to x, and for each such filter inspect its bucket to determine whether a vector y forms a neighboring pair with x.
- Iterate over each filter f in a random product code $\mathcal{C}$ and search for neighboring pairs within the corresponding bucket.

The first strategy benefits from Grover's algorithm, whereas the second strategy can rely on collision-finding techniques. The next section explains the first strategy in detail. A subsequent section introduces quantum random walks and shows how they lead to collision-finding methods suited to LSF algorithms.

Algorithm 1 Classical LSF Scheme [Laa15]

Require: A list L of vectors sampled uniformly on $\mathbb{S}^{d-1}$ of size N.
Ensure: A list L' of N distinct neighboring pairs of elements of L.

generate filters $f_1, \ldots, f_r \in \mathcal{C}$
initialize α-buckets $B_\alpha(f_1), \ldots, B_\alpha(f_r)$
$L' \leftarrow \varnothing$
for $x \in L$ **do** ▷ Initialization step
    compute $K_\alpha(x) := \mathcal{H}_{x,\alpha} \cap \mathcal{C}$
    for $f_i \in K_\alpha(x)$ **do**
        add x to $B_\alpha(f_i)$
for $x \in L$ **do** ▷ Search step
    for $f_i \in K_\alpha(x)$ **do**
        for $y \in B_\alpha(f_i), y \neq x$ **do**
            if (x, y) is a neighboring pair **then**
                add (x, y) to L'
return L'

Quantum time complexity. Consider a list L of N lattice vectors, a random product code $\mathcal{C}$ of size N^c, and an angle α with $\pi/3 \leq \alpha \leq \pi/2$. One can express the quantum time complexity T_{LSF} of a quantum LSF scheme in terms of N, c, and α:

$$T_{\text{LSF}} = N \cdot (N^c \, \mathcal{V}_d(\alpha)) + N \cdot \sqrt{N^c \, \mathcal{V}_d(\alpha) \cdot N \, \mathcal{V}_d(\alpha)}.$$

The search step enjoys a quadratic improvement over its classical counterpart (algorithm 1) thanks to Grover's algorithm, which accounts for the square root in the second term.

2.5 Filter Sampling

Filter Sampling. Heiser shows in [Hei21] that one can construct a circuit that samples new filters that are ω–close to a fixed vector x in the unit sphere $\mathbb{S}^{d-1}$. The sampling procedure runs in subexponential time.

Theorem 1 ([Hei21]). *Let $\mathcal{C}$ be a random product code with $m = \log(d)$ subcodes of size b, such that $b^m = N^{O(1)}$. Let x be a vector in $\mathbb{S}^{d-1}$ and let ω be an angle. There exists an algorithm with a preprocessing routine running in time $O(m^2 db)$ that subsequently samples a filter f_i that is ω–close to x in time $O(mb)$. Therefore this algorithm runs in time $2^{o(d)}$.*

Time complexity. Let L be a list of N lattice vectors and let $\mathcal{C}$ be a random product code containing N^c codewords. Let β and ω be two angles satisfying $\frac{\pi}{3} \leq \beta \leq \omega \leq \pi$. The running time of Algorithm 2 is

$$T_{\text{quantum}} = N \cdot (N^c \, \mathcal{V}_d(\beta)) + N \cdot \sqrt{N^c \, \mathcal{V}_d(\omega) \, N \mathcal{V}_d(\beta)}.$$

Using filter sampling decreases the original complexity of [Laa16] from $2^{0.2652d+o(d)}$ to $2^{0.2571d+o(d)}$, which is close to the exponent $2^{0.2570d+o(d)}$ obtained in [CL21].

3 Quantum Random Walk Preliminaries

As explained in the previous section, a LSF scheme consists of two main steps: an initialization step and a search step. During the search step, an effective strategy is to identify many reducing pairs within each bucket. In this section, we present a quantum walk approach for performing this search more efficiently. Quantum walks provide a general framework for finding marked vertices in graphs. By choosing a Johnson graph, the quantum walk can be used to efficiently search for collisions inside a list, and then to perform the search step.

3.1 Finding Collisions using a Quantum Walk

Quantum Random Walk in general. Consider a graph $G = (V, E)$ together with a distinguished subset of vertices, denoted by M, that we refer to as marked. Let ε be the probability that a uniformly chosen vertex from V lies in M. Let δ denote the spectral gap of the graph G.

Theorem 2 ([Mag+11]). *Let $G = (V, E)$ be a graph, let $M \subseteq V$ be the set of marked vertices, let ε be the fraction of marked vertices, and let δ be the spectral gap of G. We introduce the following costs:*

- S*, the cost of initializing the starting vertex,*
- U*, the cost of updating a vertex,*
- C*, the cost of checking whether a vertex is marked.*

A quantum random walk locates a marked vertex within time

$$T \;=\; \mathsf{S} \;+\; \frac{1}{\sqrt{\varepsilon}}\Big(\mathsf{C} \;+\; \frac{1}{\sqrt{\delta}}\,\mathsf{U}\Big).$$

Johnson graph. A quantum random walk becomes useful when one seeks a collision in a list of elements $\{y_1, \ldots, y_n\}$. The classical construction for this task uses the Johnson graph $J(\{y_1, \ldots, y_n\}, r)$. In this graph, the vertices are subsets v of $\{y_1, \ldots, y_n\}$ that contain exactly r distinct elements. Two vertices v and w are adjacent when one can obtain w from v by replacing a single element; that is, there exist elements y_{old} and y_{new} in $\{y_1, \ldots, y_n\}$ such that

$$w = \big(v \setminus \{y_{\text{old}}\}\big) \cup \{y_{\text{new}}\}.$$

A vertex v is marked when it contains two distinct elements that form a collision. The set of marked vertices is

$$M = \big\{\, v \in V \;\big|\; \text{there exist distinct } y_i, y_j \in v \text{ such that } y_i \text{ and } y_j \text{ collide} \,\big\}.$$

3.2 Quantum Walk for the Nearest-Neighbor Search

Assumption. Let f be a filter. We assume that a bucket $B_\alpha(f)$ has been created and filled with lattice vectors x_i. Chailloux and Loyer [CL21] model these vectors as uniformly distributed on the boundary of the bucket. This assumption relies on the observation that for every sufficiently small angle $\varepsilon > 0$ one has

$$\Pr\big(\alpha - \varepsilon \le \theta(x, f) \le \alpha\big) \approx \Pr\big(\theta(x, f) \le \alpha\big).$$

Reduced vector. For each vector x_i we may write

$$x_i = \cos(\alpha)\, f + \sin(\alpha)\, y_i,$$

where y_i is a unit vector in the hyperplane $f^\perp$ of dimension $d-1$. We say that y_i is the reduced form of x_i. Every y_i lies on the sphere $\mathbb{S}^{d-2}$ and the distribution of the vectors y_i is uniform. We denote by L_y the list of all reduced lattice vectors.

New neighboring conditions. The condition under which (x_i, x_j) forms a neighboring pair can be rewritten in terms of the reduced vectors. The pair (y_i, y_j) is said to be neighboring when the angle between them does not exceed

$$\theta^*_\alpha = 2 \arcsin\Big(\frac{1}{2\sin(\alpha)}\Big).$$

Our task is therefore to identify θ^*_α-close pairs inside L_y, whose elements lie uniformly on $\mathbb{S}^{d-2}$. This problem resembles an instance of the shortest-vector problem. For this reason it is advantageous to sample a new random product code $\mathcal{C}_2$, construct new filters with angle β, and fill the associated buckets in order to locate neighboring pairs more efficiently.

Quantum walk parameters. The quantum random walk is applied to detect a collision in a modified Johnson graph $J(L_y, r)$. Each vertex v is a triple $(L^v_y, D(v), b^v_y)$, where L^v_y is a subset of L_y of size r, the structure $D(v)$ stores auxiliary data that accelerates the verification procedure, and the bit b^v_y indicates whether the vertex is marked. Following [CL21], the definitions are:

- $M = \big\{\, v \in V \;\big|\; \exists\, t \in \mathcal{C}_2,\ \exists\, y_i \neq y_j \in J^v_\beta(t),\ y_i \text{ and } y_j \text{ are } \theta^*_\alpha\text{-close} \,\big\}$,
- $D(v) = \big\{\, J^v_\beta(t_i) \,\big\}_{t_i \in \mathcal{C}_2}$, where $\mathcal{C}_2 \subset \mathbb{S}^{d-2}$ is a random projection code and

$$J_\beta(t_i) = \mathcal{H}_{(\beta, t_i)} \cap \mathcal{C}_2$$

  is the corresponding β-bucket.

3.3 More Details on the Quantum Walk Update

To motivate our choice, we first explain how a vertex is updated in [CL21].

Update phase algorithm. We consider two connected vertices, denoted by v and w. The connection implies that there exist elements $y_{\text{old}} \in L_y^v$ and $y_{\text{new}} \in L_y^w$ such that

$$L_y^w = \left(L_y^v \setminus \{y_{\text{old}}\}\right) \cup \{y_{\text{new}}\}.$$

The update proceeds in two steps. First, we remove y_{old} from L_y^v and from every bucket $J_\beta^v(t)$, and then we update the corresponding bit. Second, we insert y_{new} into L_y^v and into each relevant bucket. The following pseudocode illustrates this procedure.

Algorithm 2 UPDATE_PHASE$(v, (y_{\text{old}}, y_{\text{new}}))$

Require: A vertex v of the graph. Reduced vectors $y_{\text{old}}, y_{\text{new}} \in \mathbb{S}^{d-2}$ to update.
Ensure: The vertex w, where L_y^w, $D(w)$ and b^w are update regarding $(y_{\text{old}}, y_{\text{new}})$.
1: Delete y_{old} from L_y^v, its buckets and update b^v in consequence
2: Set $w \leftarrow v$
3: Compute $K_\beta(y_{\text{new}}) := \mathcal{H}_{(\beta,y)} \cap \mathcal{C}_2$ ▷ Initialization Step
4: **for** $t \in K_\beta(y_{\text{new}})$ **do**
5: add y_{new} to $J_\beta^w(t)$
6: **for** $t \in K_\beta(y_{\text{new}})$ **do** ▷ Search Step
7: **for** $y_i \neq y_{\text{new}}$ in $J_\beta^v(t)$ **do**
8: **if** $y_i \overset{\theta_\alpha^*}{-} y_{\text{new}}$ **then**
9: update b^w rergarding y_{new}
10: **return** $w = (L_y^w, \{J_\beta^w(t)\}_{t \in \mathcal{C}_2}, b^w)$

Update cost. The routine for removing y_{old} (line 1) is almost the same as the one detailed for insterting y_{new}. The update of the boolean value b^v can be accelerated with Grover search, which yields the cost

$$\mathsf{U} = 2 \cdot \left(|\mathcal{C}_2| \cdot \mathcal{V}_d(\beta) + \sqrt{|\mathcal{C}_2| \ \mathcal{V}_d(\beta) \cdot \left|L_y^v\right| \ \mathcal{V}_d(\beta)} \right).$$

The algorithm first initializes a collection of β-buckets and then searches for a pair that is θ_α^*-close. This structure is nearly identical to the locality-sensitive filtering method of [Laa16] described in Section 1. This similarity motivates the use of filter sampling [Hei21] to improve the update phase. The goal is to construct smaller buckets, which accelerates the work performed in lines 2–3, and to inspect filters that lie farther from y in lines 4–7. Implementing this modification requires altering the marking condition M, which in turn changes several parameters of the quantum walk such as $\mathcal{S}$, U, and ε.

Time complexity parameters. We now have three free parameters $0 \leq c_2 \leq c_3 \leq c_1 \leq c$ from which all other parameters depend. In comparison to [CL21], we have added a new parameter $c_3 > c_2$ in order to add a new angle ω which will be used to put a filter sampler inside the QRW algorithm.

– $N = |L|$, lattice vectors,
– $N^c = |L| \cdot \mathcal{V}_d(\alpha) = |L_y|$, vectors per α-buckets,
– $N^{c_1} = \left|L_y^v\right|$, vectors in a vertex's list,
– $N^{c_2} = \left|L_y^v\right| \cdot \mathcal{V}_d(\beta)$, vectors per β buckets,
– $N^{c_3} = \left|L_y^v\right| \cdot \mathcal{V}_d(\omega)$, vectors ω-close to a filter $t \in \mathcal{C}_2$.

Moreover, one sets the function $\rho_0(c, c_1, c_2, c_3) > 0$ such that

$$N^{\rho_0} = \frac{\mathcal{V}_d(\beta)}{\mathcal{W}_d(\beta, \omega|\theta_\alpha^*)}.$$

Hence other auxiliary quantities can be deduced using the parameters c, c_1, c_2, c_3:

$$\mathcal{V}_d(\beta) = \frac{1}{N^{c_1-c_2}}, \mathcal{V}_d(\omega) = \frac{1}{N^{c_1-c_3}}, |\mathcal{C}_2| = \frac{1}{\mathcal{W}_d(\beta, \omega|\theta_\alpha^*)} = N^{\rho_0+c_1-c_2}, |L_y^v| = N^{c_1}.$$

3.4 Another Graph Setting

Marking condition. In this section we describe modifications of the first algorithm of [CL21] using the techniques of [Hei21]. We continue to work with a Johnson graph $J(L_y, r)$, and we adjust the vertex data $D(v)$ together with the marking condition. The new marking set is

$$M' = \{\, v \in V \mid \text{there exists } t \in \mathcal{C}_2 \text{ and } y_1, y_2 \in L_y^v \text{ such that } y_1 \overset{\beta}{-} t \overset{\omega}{-} y_2 \,\}.$$

These changes support a modified version of the checking phase $\mathcal{C}$ and the update phase $\mathcal{U}$ of the quantum random walk. When we choose an element $y \in L_y^v$ to update, we no longer search for β-close filters associated with β-buckets. Instead, we search for ω-close filters associated with β-buckets, where the parameter ω is strictly larger than β.

Vertex data. As in the previous section, we use the Johnson graph $G = J(L_y, r)$ where r is a positive integer. A vertex v of G consists of a list L_y^v and a data structure $D(v)$. The data has the form

$$D(v) = \left[D_\beta(v),\, D_\omega(v),\, L_{\text{sol}}^v\right].$$

The sets $D_\omega(v)$ and $D_\beta(v)$ are used to accelerate the update phase. The list L_{sol}^v replaces the marking bit and stores all neighboring pairs contained in L_y^v. We define these components as follows:

– $D_\omega(v) = \{\, J_\omega^v(t) \,\}_{t \in \mathcal{C}_2}$, with

$$J_\omega^v(t) = \{\, y \in L_y^v \mid \exists y' \in L_y^v, y \overset{\theta_\alpha^*}{-} y',\, y \overset{\omega}{-} t \overset{\beta}{-} y', \text{ and } \neg(y \overset{\beta}{-} t) \,\}.$$

– $D_\beta(v) = \{\, J_\beta^v(t) \,\}_{t \in \mathcal{C}_2}$, with

$$J_\beta^v(t) = L_y^v \cap \mathcal{H}_{t,\beta}.$$

After one iteration of the update and checking procedures, the resulting vertex again satisfies these properties.

Remark 2. The article [CL21] uses the truncated sets $\mathbf{J}_\beta^v(t)$ rather than the full sets $J_\beta^v(t)$, where $\mathbf{J}_\beta^v(t)$ contains only the first $2N^{c_2}$ elements of $J_\beta^v(t)$. This truncation is required to bound the update cost. Using $\mathbf{J}_\beta^v(t)$ instead of $J_\beta^v(t)$ provides a worst-case running-time bound. In the remainder of this document we write $J_\beta^v(t)$ for simplicity, and arguments similar to Lemma 2 of [CL21] ensure that this choice does not affect the asymptotic results.

Remark 3. The article [CL21] does not require an explicit list of solutions to set the marking bit. We include the list L_{sol}^v in our description because it clarifies statements such as "In which situation should we change the marking bit?" and therefore improves the readability of the algorithmic presentation.

3.5 Quantum Walk Complexity

The time complexity of the quantum random walk depends on five costs defined previously, namely the setup cost S, the update cost U, the checking cost C, and the parameters ε and δ. The checking phase requires only the verification of the last bit in a node, so the corresponding time complexity is $\mathsf{C} = 1$. The spectral gap of the Johnson graph is well-known.

Update. During the update phase, the data structures that can be updated efficiently are the list L_y^v and the data $D_\beta(v)$. Specifically, one can choose one element y_{new} to insert and one element y_{old} to remove. This allows the computation of the new vertex w such that

$$L_y^w = \left(L_y^v \setminus \{y_{\text{old}}\}\right) \cup \{y_{\text{new}}\}.$$

Consequently, all elements $J_\beta^v(t)$ for $t \in \mathcal{C}_2$ are updated. Compared to the procedure in [CL21], the main difference lies in the condition for updating the marking bit and the additional data $D_\omega(v)$.

Algorithm 3 NEW_UPDATE_PHASE$(v, y_{\text{old}}, y_{\text{new}})$

Require: A vertex v of the graph. A vector $y_{\text{old}} \in L_y^v$ and a vector $y_{\text{new}} \in L_y \backslash L_y^v$.
Ensure: The vertex w such that $L_y^w = (L_y^v \backslash \{ y_{\text{old}} \}) \cup \{ y_{\text{new}} \}$.

Set $L_y^w = \{ L_y^v \backslash \{ y_{\text{old}} \} \} \cup \{ y_{\text{new}} \}$ and $L_{\text{sol}}^w = L_{\text{sol}}^v$
Initialize $J_\beta^w(t) = J_\beta^v(t)$ and $J_\omega^w(t) = J_\omega^v(t)$ for all $t \in \mathcal{C}_2$
- Delete y_{old} from each current buckets -
Compute $K_\beta(y_{\text{old}})$
for t in $K_\beta(y_{\text{old}})$ **do**
    Delete y_{old} from $J_\beta(t)$
- Modify the marking bit regarding the loss of y_{old} -
Preprocess $K_\omega(y_{\text{old}})$
for t in $K_\omega(y_{\text{old}})$ **do**
    for $y \neq y_{\text{old}}$ in $J_\beta^w(t) \cup J_\omega^v(t)$ **do**
        if $\theta(y, y_{\text{old}}) \leq \theta_\alpha^*$ **then**
            Delete (y, y_{old}) from L_{sol}^w
            if $\theta(y, t) \geq \beta$ **then**
                Delete y from $J_\omega^w(t)$
Do similar steps considering the addition of y_{new}
- Conclusion -
$D(w) = [\{ J_\beta^w(t) \}_{t \in \mathcal{C}_2}, \{ J_\omega^w(t) \}_{t \in \mathcal{C}_2}, L_{\text{sol}}^w]$
if L_{sol}^w is empty, $b^w \leftarrow 0$, **else**, $b^w = 1$
Return $w = (L_y^w, D(w), b^w)$

Using Proposition 3, we can compute the list $K_\beta(y)$ for a given vector y in the same time required to browse the list. We perform this operation twice, once for y_{old} and once for y_{new}. Therefore, the update cost (ignoring subexponential terms) can be expressed as

$$\mathsf{U} = |K_\beta(y)| + \sqrt{|K_\omega(y)| \cdot \left(\left|J_\beta^v(t)\right| + |J_\omega^v(t)|\right)}.$$

In more general notation, these terms can be written as

- $|K_\beta(y)| = |\mathcal{C}_2| \cdot \mathcal{V}_d(\beta)$,
- $|K_\omega(y)| = |\mathcal{C}_2| \cdot \mathcal{V}_d(\omega)$,
- $\left|J_\beta^v(t)\right| = \left|L_y^v\right| \cdot \mathcal{V}_d(\beta)$,
- The quantity $|J_\omega^v(t)|$ corresponds to the average number of elements y that form a neighboring pair, multiplied by the probability that such a pair lies in a hypercone of angle ω. This gives

$$\begin{aligned} |J_\omega^v(t)| &\leq |L_{\text{sol}}^v| \cdot \mathcal{V}_d(\omega) \\ &\leq \left|L_y^v\right|^2 \cdot \mathcal{W}_d(\beta, \omega \mid \theta_\alpha^*) \cdot \mathcal{V}_d(\omega). \end{aligned}$$

Using these expressions, we can rewrite the update cost as

$$\begin{aligned}\mathsf{U} &= |\mathcal{C}_2| \cdot \mathcal{V}_d(\beta) + \sqrt{|\mathcal{C}_2| \cdot \mathcal{V}_d(\omega)\Big(\big|L_y^v\big| \cdot \mathcal{V}_d(\beta) + \big|L_y^v\big|^2 \cdot \mathcal{W}_d(\beta, \omega \mid \theta_\alpha^*) \cdot \mathcal{V}_d(\omega)\Big)} \\ &\leq N^{\max\{\rho_0, (\rho_0+c_3)/2, c_3\}}.\end{aligned}$$

Ratio of marked vertices ε. We have modified the marking condition for vertices. Consequently, the ratio of marked vertices in the graph changes. The following proposition quantifies this ratio.

Proposition 4. *Let ε denote the ratio of marked vertices. Then*

$$\varepsilon \geq \min\Big(1, |L_y|^2 \cdot \mathcal{V}_d(\theta_\alpha^*)\Big) \cdot |\mathcal{C}_2| \cdot \mathcal{W}_d(\beta, \omega \mid \theta_\alpha^*).$$

Proof. We define two events. First, let E_0 denote the event that the list L_y contains a neighboring pair (y_1, y_2). Second, let E_1 denote the event that for $y_1, y_2 \in \mathbb{S}^{d-2}$, there exists $t \in \mathcal{C}_2$ such that y_1 is β-close to t and t is ω-close to y_2. The probability of E_0 satisfies

$$\Pr(E_0) \leq \min\big(1, |L_y|^2 \cdot \mathcal{V}_d(\theta_\alpha^*)\big).$$

If t is sampled uniformly from $\mathbb{S}^{d-2}$ and y_1 is θ_α^*-close to y_2, then the probability that t lies in $\mathcal{H}_{y_1,\beta} \cap \mathcal{H}_{y_2,\omega}$ is $\mathcal{W}_d(\beta, \omega \mid \theta_\alpha^*)$. Since the elements of $\mathcal{C}_2$ are independent and uniformly distributed, we obtain

$$\Pr(E_1 \mid E_0) \leq |\mathcal{C}_2| \cdot \mathcal{W}_d(\beta, \omega \mid \theta_\alpha^*).$$

Using the chain rule of probability, we compute

$$\begin{aligned}\varepsilon &= \Pr(E_0 \cap E_1) \\ &= \Pr(E_0)\ \Pr(E_1 \mid E_0) \\ &\leq \min\Big(1, |L_y|^2 \cdot \mathcal{V}_d(\theta_\alpha^*)\Big) \cdot |\mathcal{C}_2| \cdot \mathcal{W}_d(\beta, \omega \mid \theta_\alpha^*).\end{aligned}$$

Remark 4. Following [CL21], we first choose $|\mathcal{C}_2|$ such that $|\mathcal{C}_2| \cdot \mathcal{W}_d(\beta, \omega \mid \theta_\alpha^*) = 1$ to counterbalance the factor $\mathcal{W}_d(\beta, \omega \mid \theta_\alpha^*)$. We will later study the case where $|\mathcal{C}_2|$ is not fixed.

Setup. Setting up a vertex v consists of initializing the list L_y^v, constructing all buckets $J_\beta^v(t)$ and $J_\omega^v(t)$, and determining the marking bit b^v. The main cost comes from constructing $J_\beta^v(t) = \mathcal{H}_{t,\beta} \cap L_y^v$. We can initialize $J_\beta(t)$ and compute

$$K_\beta(y_i) = \mathcal{H}_{y_i,\beta} \cap \mathcal{C}_2$$

for each $y_i \in L_y^v$. Then, if $t \in K_\beta(y_i)$, we add y_i to $J_\beta^v(t)$. To initialize the marking bit, we apply Grover's algorithm over all pairs in L_y^v to find a neighboring pair (y_1, y_2). This step requires time $\sqrt{\big|L_y^v\big|^2} = \big|L_y^v\big|$. Since $|\mathcal{C}_2| \cdot \mathcal{W}_d(\beta, \omega \mid \theta_\alpha^*) = 1$, there is on average one $t \in \mathcal{C}_2$ such that y_1 is β-close to t and t is ω-close to y_2. If t is not β-close, we add y_2 to $J_\omega^v(t)$. Overall, the setup cost is

$$\begin{aligned}\mathsf{S} &= \big|L_y^v\big| \cdot |K_\beta(y)| + \big|L_y^v\big| \\ &= N^{\max\{\, c_1+\rho_0, c_1\,\}}.\end{aligned}$$

3.6 New Total Time for QRW for LSF Scheme

We denote by r the size of the set L_y^v and by s the size of the set $\mathcal{C}_2$. The quantum time complexity of the quantum random walk with filter sampling, given the parameters (c, c_1, c_2, c_3), is

$$\begin{aligned} T_{\mathrm{QRW}}^{(c,c_1,c_2,c_3)} &= \mathsf{S} + \frac{1}{\sqrt{\varepsilon}}\left(\mathsf{C} + \frac{1}{\sqrt{\delta}}\,\mathsf{U}\right) \\ &= r \cdot s \cdot \mathcal{V}_d(\beta) + \frac{1}{\sqrt{\min\{1,\, r^2 \cdot \mathcal{V}_d(\theta_\alpha^*)\}}}\left(1 + \frac{2}{\sqrt{r}}\left(s \cdot \mathcal{V}_d(\beta) + \sqrt{s\mathcal{V}_d(\omega) \cdot r\mathcal{V}_d(\beta)}\right)\right) \\ &= N^{c_1+\rho_0} + \frac{1}{\min\left(1,\, N^{c_1 - \frac{\rho_0 + c_1 - c_2}{2}}\right)}\left(1 + N^{\frac{c_1}{2} + \max\{\rho_0,\, c_3\}}\right). \end{aligned}$$

The main modification consists in replacing the coefficient c_2 with c_3 in the term U and in making ρ_0 dependent on c_3. When c_2 equals c_3, the expression $T_{\mathrm{QRW}}^{(c,c_1,c_2,c_2)}$ coincides with the quantum time complexity of the quantum random walk used in [CL21].

3.7 Numerical Results

We refer to [CL21] for the complete formula of the total cost of the lattice sieving algorithm using local sensitive filtering via quantum random walk. The total cost, as a function of the parameters (c, c_1, c_2, c_3), is

$$T_{\mathrm{total}}^{(c,c_1,c_2,c_3)} = N_{\mathrm{REP}}\Big[T_{\mathrm{INIT}} + N_{\mathrm{REP2}} T_{\mathrm{QRW}}\Big],$$

where the quantities N_{REP}, T_{INIT}, and N_{REP2} depend only on c:

- $N_{\mathrm{REP}} = \max\{\,1, N^{c-\zeta(c)+o(1)}\,\}$, with $N^{\zeta(c)} = N^{2c}\mathcal{V}_d(\theta_\alpha^*)$,
- $T_{\mathrm{INIT}} = N^{1+o(1)}$,
- $N_{\mathrm{REP2}} = N^{1-c+\zeta(c)}$.

We can therefore solve an optimization problem to determine the parameters c, c_1, c_2, c_3 that minimize T_{total}. For this purpose, we use the `scipy.optimize` package in Sage. The results of this optimization are reported in the table below.

Table 1. Numerical optimized results

	Total Time	c	c_1	c_2	c_3
[CL21] – algo1	0.2605	0.3300	0.1972	0.0615	0.0615
algo1 - enhanced	0.2594	0.3426	0.2425	0.0003	0.05440

The numerical results confirm that the filter sampling method provides an improvement. We achieve a smaller total time by using smaller β-buckets and by searching for more distant ω-close filters. This behavior is reflected in our algorithm by having a smaller value of c_2 and a larger value of c_3. We expect that this improvement can be adapted to more efficient algorithms, such as the second algorithm in [CL21] and the algorithm in [Bon+22].

4 Application to Faster Algorithms and Disillusionment

4.1 Some Changes

The second algorithm presented in [CL21] (Theorem 4) and the algorithm of [Bon+22] are faster than our current algorithm. Both algorithms exploit the observation from Proposition 4 that the performance is not significantly affected if the size of the set $\mathcal{C}_2$ does not depend on $\mathcal{W}_d(\beta, \omega \mid \theta_\alpha^*)$. To compensate, one can increase the number of repetitions in the quantum random walk, denoted N_{REP2}. We introduce a new parameter ρ satisfying $0 \leq \rho \leq \rho_0(c, c_1, c_2, c_3)$ and define

$$|\mathcal{C}_2| = N^{\rho + c_1 - c_2}.$$

The ratio ε then becomes

$$\varepsilon = \min\bigl(1, N^{2c_1} \mathcal{V}_d(\theta_\alpha^*)\bigr) \cdot N^{\rho - \rho_0}.$$

We can now run the quantum random walk using the parameter ρ instead of $\rho_0(c, c_1, c_2, c_3)$. Because the size of $\mathcal{C}_2$ is smaller, the expected number of collisions decreases by a factor of $N^{\rho - \rho_0}$. Therefore, one must resample $\mathcal{C}_2$ multiple times to achieve the optimal number of collisions in a single α-bucket. Section 6 of [CL21] explains the modifications to the locality-sensitive filtering algorithm in detail.

4.2 Numerical Results

We solve the optimization problem associated with the total running time of the locality-sensitive filtering algorithm for the parameters c, c_1, c_2, c_3, and ρ. Previous expressions of the total time in [CL21] and [Bon+22] do not include the parameter c_3. In order to recover the same formula, one must replace the previous instance of c_2 accordingly.

Table 2. Numerical optimized results for better algorithms

	Total Time	c	c_1	c_2	c_3	ρ
[CL21] – algo2[1]	0.2570	0.3696	0.2384	0	0	0
[CL21] – algo2 – enhanced	0.2570	0.3696	0.2384	0	0	0
[Bon+22]	0.2563	0.3875	0.27	0	0	0
[Bon+22] – enhanced	0.2563	0.3875	0.27	0	0	0

The numerical results indicate that the proposed enhancements using filter sampling do not affect the final outcome. This occurs because the total time is minimized when $c_2 = c_3 = 0$, yielding the same result as the original algorithm. Moreover, inserting the optimal values of c_3 and ρ into the expression of the update cost gives $\mathsf{U} = 1$, which implies that the update phase is already optimal. This improvement was not achieved in the first algorithm described in [CL21].

4.3 Conclusion

Table 3. Quantum time complexity

	[Laa16]	[Hei21]	[CL21] – algo1	algo1 – enhanced	[CL21] – algo2	[Bon+22]
Time	0.2652	0.2571	0.2605	0.2594	0.2570	0.2563

We investigated whether Heiser's filter sampling technique [Hei21] could improve the update phase of the quantum random walk sieve of Chailloux and Loyer [CL21]. We obtained a modest improvement for the first algorithm of [CL21], as summarized in the table above. However, numerical optimization consistently yields $c_2 = c_3$ for the faster algorithms of [CL21] and [Bon+22], since their update phase is already optimal and filter sampling provides no additional gain.

Natural directions for future work include extending this approach to the quantum tuple sieve of [Eng+25], where the richer LSF structure may offer more possibilities of improvement, and investigating the existence of quantum random walk algorithms for 3-sieve variants.

References

[Kan83] Ravi Kannan. "Improved algorithms for integer programming and related lattice problems". In: *Proceedings of the fifteenth annual ACM symposium on Theory of computing*. STOC '83. New York, NY, USA: Association for Computing Machinery, 1983, pp. 193–206. ISBN: 978-0-89791-099-6. DOI: 10.1145/800061.808749. URL: https://dl.acm.org/doi/10.1145/800061.808749 (visited on 12/05/2025).

[Ajt96] Miklos Ajtai. *Generating Hard Instances of Lattice Problems*. TR96-007. ISSN: 1433-8092. Electronic Colloquium on Computational Complexity (ECCC), Jan. 30, 1996. URL: https://eccc.weizmann.ac.il/eccc-reports/1996/TR96-007/index.html (visited on 12/03/2025).

[IM98] Piotr Indyk and Rajeev Motwani. "Approximate nearest neighbors: towards removing the curse of dimensionality". In: *Proceedings of the thirtieth annual ACM symposium on Theory of computing*. STOC '98. New York, NY, USA: Association for Computing Machinery, 1998, pp. 604–613. ISBN: 978-0-89791-962-3. DOI: 10.1145/276698.276876. URL: https://dl.acm.org/doi/10.1145/276698.276876 (visited on 12/05/2025).

[NV08] Phong Q. Nguyen and Thomas Vidick. "Sieve algorithms for the shortest vector problem are practical". In: *Journal of Mathematical Cryptology*. Vol. 2. ISSN: 1862-2976, 1862-2984 Issue: 2. Jan. 2008. DOI: 10.1515/JMC.2008.009. URL: https://www.degruyter.com/document/doi/10.1515/JMC.2008.009/html (visited on 12/03/2025).

[Gen09] Craig Gentry. "Fully homomorphic encryption using ideal lattices". In: *Proceedings of the forty-first annual ACM symposium on Theory of computing*. STOC '09. New York, NY, USA: Association for Computing Machinery, 2009, pp. 169–178. ISBN: 978-1-60558-506-2. DOI: 10.1145/1536414.1536440. URL: https://dl.acm.org/doi/10.1145/1536414.1536440 (visited on 12/03/2025).

[MV10] Daniele Micciancio and Panagiotis Voulgaris. "Faster exponential time algorithms for the shortest vector problem". In: *Proceedings of the twenty-first annual ACM-SIAM symposium on Discrete algorithms*. SODA '10. USA: Society for Industrial and Applied Mathematics, 2010, pp. 1468–1480. ISBN: 978-0-89871-698-6. (Visited on 12/03/2025).

[Mag+11] Frédéric Magniez et al. "Search via Quantum Walk". In: *SIAM Journal on Computing* 40.1 (Jan. 2011), pp. 142–164. ISSN: 0097-5397, 1095-7111. DOI: 10.1137/090745854. arXiv: quant-ph/0608026. URL: http://arxiv.org/abs/quant-ph/0608026 (visited on 09/10/2025).

[Bec+15] Anja Becker et al. "New directions in nearest neighbor searching with applications to lattice sieving". In: 2015/1128 (2015). Publication info: Published elsewhere. SODA'16. URL: https://eprint.iacr.org/2015/1128 (visited on 12/03/2025).

[Laa15] Thijs Laarhoven. "Sieving for Shortest Vectors in Lattices Using Angular Locality-Sensitive Hashing". In: *Advances in Cryptology – CRYPTO 2015*. Berlin, Heidelberg: Springer-Verlag, 2015, pp. 3–22. ISBN: 978-3-662-47988-9. DOI: 10.1007/978-3-662-47989-6_1. URL: https://doi.org/10.1007/978-3-662-47989-6_1 (visited on 12/05/2025).

[BLS16] Shi Bai, Thijs Laarhoven, and Damien Stehle. "Tuple lattice sieving". In: 2016/713 (2016). Publication info: Published elsewhere. Algorithmic Number Theory Symposium (ANTS-XII) 2016. URL: https://eprint.iacr.org/2016/713 (visited on 12/03/2025).

[Laa16] Thijs Laarhoven. "Search problems in cryptography". In: (2016).

[BMW19] Shi Bai, Shaun Miller, and Weiqiang Wen. "A refined analysis of the cost for solving LWE via uSVP". In: (2019). URL: https://eprint.iacr.org/2019/502.

[CL21] André Chailloux and Johanna Loyer. "Lattice sieving via quantum random walks". In: arXiv:2105.05608 (May 12, 2021). DOI: 10.48550/arXiv.2105.05608. arXiv: 2105.05608[quant-ph]. URL: http://arxiv.org/abs/2105.05608 (visited on 09/08/2025).

[Hei21] Max Heiser. “Improved Quantum Hypercone Locality Sensitive Filtering in Lattice Sieving”. In: 2021/1295 (2021). Publication info: Preprint. MINOR revision. URL: https://eprint.iacr.org/2021/1295 (visited on 09/08/2025).

[Bon+22] Xavier Bonnetain et al. “Finding many Collisions via Reusable Quantum Walks”. In: 2022/676 (2022). Publication info: A major revision of an IACR publication in EUROCRYPT 2023. URL: https://eprint.iacr.org/2022/676 (visited on 10/08/2025).

[Eng+25] Lynn Engelberts et al. “An Improved Quantum Algorithm for 3-Tuple Lattice Sieving”. In: (2025). arXiv: 2510.08473 [quant-ph]. URL: https://arxiv.org/abs/2510.08473.