IEEE Transactions on Quantum Engineering

Received 6 June 2024; accepted 6 November 2024; date of publication 18 November 2024;
date of current version 17 December 2024.

Digital Object Identifier 10.1109/TQE.2024.3501683

Grover's Oracle for the Shortest Vector Problem and Its Application in Hybrid Classical–Quantum Solvers

MILOŠ PROKOP[1,2], PETROS WALLDEN[3], AND DAVID JOSEPH[2]

[1]School of Informatics, University of Edinburgh, EH8 9AB Edinburgh, U.K.
[2]SandboxAQ, Palo Alto, CA 94301 USA
[3]School of Informatics, University of Edinburgh, EH8 9AB Edinburgh, U.K.

Corresponding author: Miloš Prokop (e-mail: m.prokop@sms.ed.ac.uk).

The work of Petros Wallden was supported in part by the Engineering and Physical Sciences Research Council (EPSRC) under Grant EP/T001062/1, Grant EP/X026167/1, and Grant EP/T026715/1, in part by the Science and Technology Facilities Council under Grant ST/W006537/1, and in part by Edinburgh–Rice Strategic Collaboration Awards. The work of Miloš Prokop was supported by the EPSRC through Doctoral Training Partnership Studentship under Grant EP/T517811/1.

ABSTRACT Finding the shortest vector in a lattice is a problem that is believed to be hard both for classical and quantum computers. Many major postquantum secure cryptosystems base their security on the hardness of the shortest vector problem (SVP) (Moody, 2023). Finding the best classical, quantum, or hybrid classical–quantum algorithms for the SVP is necessary to select cryptosystem parameters that offer a sufficient level of security. Grover's search quantum algorithm provides a generic quadratic speedup, given access to an oracle implementing some function, which describes when a solution is found. In this article, we provide concrete implementation of such an oracle for the SVP. We define the circuit and evaluate costs in terms of the number of qubits, the number of gates, depth, and T-quantum cost. We then analyze how to combine Grover's quantum search for small SVP instances with state-of-the-art classical solvers that use well-known algorithms, such as the block Korkine Zolotorev (Schnorr and Euchner, 1994), where the former is used as a subroutine. This could enable solving larger instances of SVP with higher probability than classical state-of-the-art records, but still very far from posing any threat to cryptosystems being considered for standardization. Depending on the technology available, there is a spectrum of tradeoffs in creating this combination.

INDEX TERMS Grover's search, postquantum cryptography (PQC), quantum algorithm, quantum circuit, shortest vector problem (SVP).

I. INTRODUCTION

The most powerful known quantum algorithm, of Shor [3], is renowned for its implications for public key cryptography. The ability to factor integers efficiently fatally undermines the security of cryptosystems, such as Rivest–Shamir–Adleman [4], elliptic curve cryptography [5], and Diffie–Hellman key exchange [6], which together form a basis for authentication and secure key exchange over the Internet.

While Shor's algorithm has been known for around 30 years, the recent surge of progress in quantum computing hardware and engineering has motivated a number of works analyzing the resource requirements of implementing such algorithms. There are often different circuits and subroutines that perform the same operation in different ways, variously requiring more auxiliary qubits or higher depth, and the impact of using different subroutines must be analyzed. The effect of error correction and noisy qubits is also considered. The quantum computing hardware race is still wide open, with seven distinct technological approaches, so it is impossible to even know which platform (and hence with which optimizations) the first crypto-cracking algorithms will be run on.

In response to the quantum threat, the cryptography community has devised a new set of cryptographic primitives known as postquantum cryptography (PQC), which we expect to remain resilient against quantum attacks. However, it is generally acknowledged that going forward, parameter sets for all cryptographic algorithms will have to take into account quantum attacks (usually variations of Grover search), and thus, designs aiming for a set security level must work backward to understand the quantum circuit complexity that an attacker would require to break the cryptosystem.

One of the key families of PQC is based on lattice problems (versus integer factorization, which is used presently). Three of four algorithms recently announced to be standardized by the National Institute for Standards and Technology (NIST) are based on lattice constructions, and one of the hard problems that must remain hard in order for lattice cryptography to remain secure is called the shortest vector problem (SVP). The security of SVP has been widely investigated from a classical perspective, but quantum investigations are less advanced. On the more theoretical end, quantum tree search algorithms have been applied to enumeration techniques, while less success has been achieved applying quantum methods to sieving. On the other hand, "full-fat" quantum SVP algorithms have been proposed based on encoding SVP to the ground state of a Hamiltonian, and this has been investigated in adiabatic settings [7], on quantum annealers [8], and in the gate model [9], [10]. The drawback of some of these quantum-native approaches is that time-complexity results are far scarcer for this creed of quantum algorithms and is difficult to quantify the concrete cost for instances far from those that can realistically implemented or simulated currently.

As stressed earlier, many quantum attacks use different types of search subroutines that are all based on Grover's search algorithm. Since Grover's algorithm, fully or as part of larger hybrid algorithms, appears in all these cases, it is important to analyze the exact cost of implementing Grover's algorithm for SVP, to quantify the concrete cost and enable calculating the security level that postquantum cryptosystems offer. Implementing Grover's search for SVP efficiently can form a module that could potentially be used in a variety of existing and new attacks. Specifically, building on the Hamiltonian approaches to SVP, one can reduce the concrete resource cost.

A. OUR CONTRIBUTIONS

In this contribution, we aim to implement efficiently and modularly the Grover's oracle for SVP and then estimate its costs and some direct implications it has within larger algorithms.

1) We implement Grover's oracle for the SVP, giving a detailed analysis of the circuit design.
2) We estimate the resources required for the oracle, including the space and time complexities, the number of gates as well as the number of T-gates.
3) Given our oracle implementation, we estimate the resources required for running the full Grover's algorithm in order to solve SVP for different lattice dimensions.
4) We discuss the implications of our analysis for improving the performance of the classical, state-of-the-art, block Korkine–Zolotorev (BKZ) algorithm for solving SVP. Specifically, we use Grover's search as subroutine within BKZ and discuss the improvements one could get in terms of accuracy or running time for solving as large dimension as possible of the SVP and comment on the relevance for existing postquantum secure cryptosystems.

Here, it is important to note that in our calculations we assume perfect (noise-less) qubits. In implementing those algorithms in realistic noisy setting, overheads due to quantum error correction would be needed. These overheads depend on many things, including on the quantum error correcting codes used, noise level and other characteristics of the hardware, etc. Our results can easily be reinterpreted for all these settings by considering the gates as "fault-tolerant" gates (and their corresponding times) and similarly the number of qubits as the number of "logical" qubits. To simplify such uses of our results, we also provide the T-gate count, since those gates would have different cost when implemented fault-tolerantly.

B. RELATED WORK

Many quantum algorithms that include an oracle have been proposed, e.g., Grover's algorithm [11], Deutsch–Jozsa algorithm [12]. Shor's algorithm [3], or for the direct application to solving the SVP, the Montanaro's backtracking algorithm [13]. This oracle is a function that can be easily described by an analytical formula while a practical circuit implementation is not being considered. This facilitates high-level analysis of the algorithms omitting the essential practical implementation details. Several works have discussed approaches to implement the quantum oracles. Zhao [14] discusses small-scale implementation of the quantum oracles for the first three aforementioned algorithms, Henderson et al. [15] propose an approach for automatic synthesis of the quantum oracles for Grover's and Shor's algorithms, Sanchez-Rivero et al. [16] propose a circuit for Grover's algorithm to find elements less than a certain value in an unstructured database. Grassl et al. [17] implement Grover's oracles different key sizes of Advanced Encryption Standard (AES). They decompose the circuits into Clifford+T set and argue about qubit and depth requirements.

In the security arena, researchers have begun to investigate constructions for symmetric cryptography. To evaluate the complexity of breaking the AES, resource estimates were calculated for building an oracle to perform exhaustive key search [17]. For hash functions, oracle constructions revealed that due to the varying dependencies on addition or multiplication, SHA2 may remain harder for quantum computers to break than its successor, SHA3 [18]. For SVP specifically, constructions have shown how to encode the problem as a Hamiltonian and analyzed its performance in algorithms ranging from adiabatic quantum computation [7] to variational quantum eigensolvers [10], [19] and annealers [8]. More recently, one independent work has found lower bounds on circuit depth while searching for short vectors via enumeration with extreme pruning [20]. Another gives a concrete implementation of a circuit for enumeration using Montanaro's quantum tree backtracking algorithm [21]. It would be expected that Montanaro's speedup applied to state-of-the-art enumeration techniques would be an optimal approach; however, concrete resource requirements for certain lattice dimensions are not provided for comparison. Furthermore, there is utility to understanding traditional Grover mappings and speedups across problems as it provides a kind of benchmarking, which is not possible with more bespoke algorithms. Ultimately, the resource requirements for Grover search for short vectors via this family of Hamiltonians is high. These Hamiltonians have been studied in the case of annealing, QAOA, and adiabatic evolution [7], [8], [19], and the high resource costs indicate that these approaches may offer a natural advantage in practice over gate-based architectures which require fault tolerance and costly decompositions to universal gate sets.

C. STRUCTURE OF THIS ARTICLE

The rest of this article is organized as follows. In Section II, we give the necessary background, namely, we introduce the SVP, Grover's search algorithm, and the BKZ algorithm for solving SVP. In Section III, we define the figures of merit that we will use when evaluating the resources required for (sub)algorithms. In Section IV, we give an implementation of Grover's oracle suited for the SVP. Section V analyzes resource requirements of the single oracle circuit, and Section V-C further discusses the overall resource requirements for the full Grover's search run for different lattice dimensions up to 100. In Section VI, we analyze how using Grover to solve SVP for relatively smaller dimensions can also be important, by using it as a subroutine to improve the classical BKZ algorithm [22], and give the potential improvements. Finally, Section VII concludes this article.

II. PRELIMINARIES

A. LATTICES

A mathematical lattice is a repeating pattern of points in $\mathbb{R}^n$. It can be described by a single basis vector $\mathbf{b}_i$ in each dimension, hence n linearly independent vectors. The basis vectors taken together are known as a basis $\mathbf{B} = (\mathbf{b}_1, \ldots, \mathbf{b}_n)$. There are infinitely many bases that can be used to describe a single lattice, and each is related to each other by a unimodular transformation $\mathbf{B}' = \mathbf{U}\mathbf{B}$. Some bases contain short vectors, which are close to orthogonal, said to be good, and others contain only very long vectors. Finding short bases can be used as an intermediate step to find short vectors, and vice versa. Of a particular importance for us will be Gaussian heuristics, which is an estimate on the length of the shortest nonzero vector of the full-rank lattice $\mathcal{L}$

$$\mathrm{gh}(\mathcal{L}) = \sqrt{n/2\pi e}\mathrm{Vol}(\mathcal{L})^{1/n}$$

and Gaussian orthogonalization of lattices. Denote the orthogonal projection $\pi_i : \mathbb{R}^n \to (\mathbf{b}_1, \ldots, \mathbf{b}_{i-1})^{\perp}$. Write $\pi_i(\mathbf{B}_{[i:j]}) = (\pi_i(\mathbf{b}_1), \ldots, \pi_i(\mathbf{b}_{j-1}))$ and let the corresponding lattice be known by $\pi_i(\mathcal{L}_{[i:j)})$. Let $\mathbf{B}^* = (\mathbf{b}_1^*, \ldots, \mathbf{b}_n^*)$ denote the Gram–Schmidt orthogonalization of the basis. Then, $\pi_i(\mathbf{b}_i) = \mathbf{b}_i^*$.

B. SHORTEST VECTOR PROBLEM

One of the fundamental hard problems in the study of mathematical lattices is how to find the shortest vector, known as the SVP.

Definition 1: Given a basis $\mathbf{B}$ that describes a lattice $\mathcal{L}(\mathbf{B})$, find the shortest vector $\mathbf{v} \in \mathcal{L}$, which has length $\|\mathbf{v}\| = \lambda_1(\mathcal{L})$.

A more relaxed variant of SVP is called γ-approximate SVP, or SVP$_\gamma$. This requires the solver to find a vector $\mathbf{v} \in \mathcal{L}$ such that $\|\mathbf{v}\| \leq \gamma \cdot \lambda_1$ where $\gamma = \mathrm{poly}(n)$. It is this more relaxed version that seeks solutions in this work. We note that, technically, the zero vector $\mathbf{0}$ is a vector in all lattices by definition. However, this trivial solution is not a permissible SVP solution. This is an important when constructing quantum Hamiltonians describing SVP solutions in the following.

C. SVP ALGORITHMS

There are two approaches to solving SVP. The first is sieving, where one samples many vectors from a somewhat short distribution and iteratively combines them until probabilistically reaching a short vector. The other is enumeration where one effectively counts all the vectors inside a ball of a determined size deterministically.

Classically, the cost of the enumeration subroutine is $2^{O(n \log n)}$[1] on an n-dimensional lattice, which, for our BKZ purposes, becomes β. Lattice sieving runs faster, taking $2^{0.292\beta+o(\beta)}$ time, though it requires exponential space. Typically, enumeration outperforms sieving for small problem sizes until a crossover point around at around dimension 70 [23]. That is to say, despite better asymptotic performance, enumeration is still a better choice for some practical choices of n.

D. GROVER'S ALGORITHM

Grover's algorithm [11] offers a quadratic advantage in time complexity for search in an unstructured database of N elements. The solution space of size M is defined by the quantum oracle O_f with action $O_f|x>y = |x>q \oplus f(x)$, where $f(x) : \{0, 1\}^n \to \{0, 1\}$ is equal to 1 if and only if x is in the solution space. The time complexity of the algorithm is then $O(\sqrt{N/M})$. As shown in Fig. 1, it consists of an iterative application of a Grover step $\hat{G}$ (see Fig. 2), where each Grover step consists of a single application of the Oracle O_f followed by a Grover reflection $H^{\otimes n}(\hat{M}_0 := 2|0><0| - 1)H^{\otimes n}$. To construct a Grover's oracle, one needs to find a quantum circuit that implements the oracle O_f. Since it is to be reused once for each of the Grover's iterations, its action

[1]Unless a base of a logarithm is specifically stated, in this article, we consider $\log(\cdot)$ to be a logarithm with base 2.

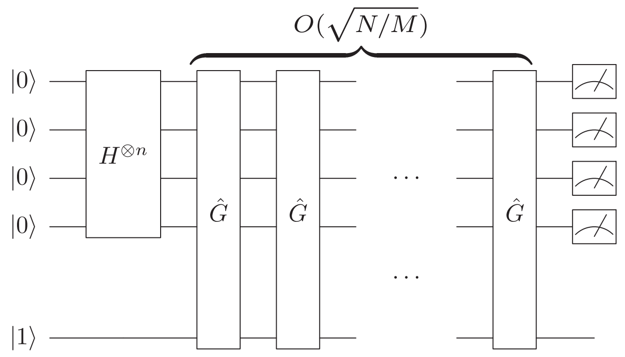

FIGURE 1. **Grover's search algorithm.**

on the input qubit register $|x>$ must be uncomputed once a conditional flip on the target qubit is performed.

E. BKZ ALGORITHM

One of the central disciplines in the study of lattices is basis reduction. This consists of iteratively reducing the length of basis vectors, with algorithms taking as input one basis and returning another basis of lattice vectors according to some kind of internal logic. The original basis reduction algorithm is the Lenstra–Lenstra–Lovász (LLL) algorithm [24], but the most commonly used in classical cryptanalysis is the BKZ algorithm [22], [25], [26], in which blocks of basis vectors (spanning a sublattice) are reduced with respect to one another, and then, the reduction is repeated with a new block of basis vectors.

Definition 2: Let $\mathbf{B} = (\mathbf{b}_1, \ldots, \mathbf{b}_n)$ define the lattice $\mathcal{L}(\mathbf{B})$. Let $\mu_{ij} := \langle \mathbf{b}_i, \mathbf{b}_j^* \rangle / \langle \mathbf{b}_j^*, \mathbf{b}_j^* \rangle$. Then, $\mathbf{B}$ is *size reduced* if $\mu_{ij} \leq 1/2$ for all $i \geq j$, with $i \leq n$. Then, $\mathbf{B}$ is said to be Hermite–Korkine–Zolotarev (HKZ) *reduced* if it is both size reduced and satisfies $\|\mathbf{b}_i^*\| = \lambda_1(\pi_i(\mathcal{L}_{[i,n)}))$. Furthermore, it is said to be BKZ-β reduced if it

1) is size-reduced;
2) satisfies $\|\mathbf{b}_i^*\| = \lambda_1(\pi_i(\mathcal{L}_{[i,\min(i+\beta-1,n)]}))$

where β is the block size.

The BKZ algorithm can be thought of as an interpolation between the LLL algorithm, where neighboring vectors in the basis are reduced pairwise and then reordered, and HKZ where the block size is n. BKZ [22] takes as an input block size β and searches for shortest vectors in the projected lattices of rank $\leq \beta$. It does this by first reducing the lattice spanned by the first β vectors and then the window of vectors moves along by one, until the window reaches the final vector in the basis. Then, the window decreases in size until only the final two vectors are reduced. This is known as one "tour," in which all vectors are updated, and $n-1$ calls to the SVP oracle are made. It was shown in [27] that within $\Theta(\frac{n^2}{\beta^2} \log n)$ tours, the first basis vector is short, with Euclidean length less than

$$\gamma_\beta^{\frac{n-1}{2(\beta-1)} + \frac{\beta(\beta-2)}{2n(\beta-1)}} \mathsf{vol}(\mathcal{L})^{1/n}$$

where γ_β and $\mathsf{vol}(\mathcal{L})$ are the Hermite constants and the volume of the lattice, respectively.

Experiments have suggested two typical uses of BKZ.

1) $\beta \approx 20$ for a small n ($<\approx 80$) or $30 \leq \beta \leq 40$ for medium dimension of n (≈ 100). Using these settings BKZ terminates in a reasonable time and tends to improve the quality of LLL-reduced basis.
2) For a high dimension n, it is chosen $\beta \geq 40$, and practically, the computation needs to be terminated early as the runtime is too long. Hanrot et al. [26] argue that reasonable early termination still outputs highly reduced bases, which is only slightly worse than the full BKZ run. The authors show that if the β-HKZ reduction (or SVP subroutine) is run $\Omega(\frac{2^3}{\beta^2}(\log n + \log\log \max_i |b_i|))$ times, then BKZ returns a basis with a norm $\leq 2\nu_\beta^{\frac{n-1}{2(\beta-1)}+\frac{3}{2}}$.

The block size β parameterizes this interpolation, and larger β results in shorter vectors but at higher computation cost. Currently, using improvements introduced in BKZ2.0, block sizes of up to dimension 120 are solvable [22]. To subvert lattice security, we would need to perform BKZ with block sizes of around 400, with Kyber-512 requiring block size 406, DiLithium requiring 423, and Falcon-512 needing 411.

F. HAMILTONIAN APPROACHES TO SVP

Previous works [7], [8], [9], [10], [19] have investigated quantum variational routines to solve SVP. Central to these works is the definition of a Hamiltonian, which encodes the length of lattice vectors into the eigenenergies of the Hamiltonian. Upon measurement of the final state, the configuration that qubits assume corresponds to an eigenstate that can be interpreted as a lattice vector, and the corresponding eigenenergy of that vector is its length. Thus, methods that find low energy eigenstates in this context find short length vectors.

The reader wants to know the combination of basis vectors that will give a short vector. So, we take the description of a lattice vector $\mathbf{v} = \mathbf{x}\mathbf{B}$, where $\mathbf{x}$ is the coefficient vector, which is read off from the qubit configuration. The Euclidean length of the vector is $\sqrt{\mathbf{v}\mathbf{v}^T}$, but the square root does not affect the ordering of vectors by length, so the authors ignore it. Thus, the square of the Euclidean length of a general lattice vector can be written as a combination of the input basis vectors as

$$\|\mathbf{v}\|^2 = \mathbf{x}\mathbf{B}(\mathbf{x}\mathbf{B})^T = \mathbf{x}\mathbf{B}\mathbf{B}^\mathbf{T}\mathbf{x}^\mathbf{T}. \quad (1)$$

Next, the substitution $\mathbf{B}\mathbf{B}^\mathbf{T} = \mathbf{G}$ is used, where $\mathbf{G}$ is known as the Gram matrix. The aim is to find an optimal $\mathbf{x}$, so the optimization occurs over the coordinates of $(x_1, \ldots, x_n)$.

To turn the expression (1) into a quantum Hamiltonian, the coordinates x_i are replaced by qudit operators $\hat{Q}^{(i)}$, where a qudit is a $2d$-level quantum system such that when measured, the output is an integer in the range $[-d+1, d]$. Applying

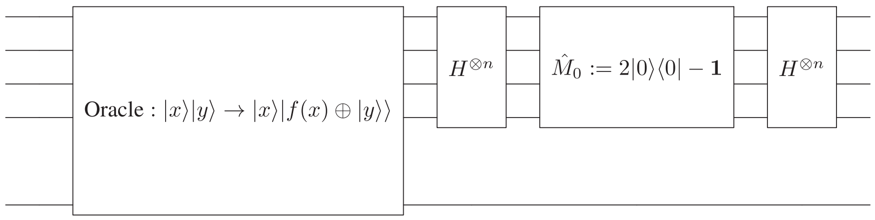
Oracle : $|x\rangle|y\rangle \rightarrow |x\rangle|f(x) \oplus |y\rangle\rangle$
$H^{\otimes n}$
$\hat{M}_0 := 2|0\rangle\langle 0| - \mathbf{1}$
$H^{\otimes n}$

FIGURE 2. Single Grover iteration $\hat{G}$.

this transformation, the Hamiltonian can then be written as

$$\hat{H} = \sum_{i,j} \hat{Q}^{(i)}\hat{Q}^{(j)}\mathbf{G}_{ij} \tag{2}$$

where $\hat{Q}^{(i)}$ represents a qudit operator on qudit i, the readout of which is a value for x_i. The eigenenergies of the Hamiltonian are the vector lengths squared for all 2^N vectors $\mathbf{v}$, which are possible configurations of N qubits. The Hamiltonian can be degenerate, as multiple vectors can have the same length. A notion of successive minima is often used to order the lattice vectors by their length: $\lambda_i(\mathcal{L})$ is the length of the ith shortest nonzero vector of $\mathcal{L}$ while degeneracies are being ignored (i.e., $\lambda_2(\mathcal{L})$ is length of the second shortest nonzero vector in $\mathcal{L}$ no matter how many lattice vectors of length $\lambda_1(\mathcal{L})$ there exist). The goal of the SVP is finding a vector in $\mathcal{L}$ of length $\lambda_1(\mathcal{L})$. In order to select the number of qubits required to implement the Hamiltonian of 2, multiple quantities may be utilized. Gaussian heuristics give an estimate on the length of $\lambda_1(\mathcal{L})$, whereas Minkowski's bound [28] bounds this quantity. This information may be used to find reasonable values of plausible ranges of the coefficients $\{x_i\}_{1\leq i\leq n}$. We denote these bounds as $|x_i| \leq d_i$, and the strategy to find their values is discussed with greater detail in [9], where it has also been proved that, in average, the scaling of $d_i = \lceil \log_2 n \rceil$ is sufficient, and we therefore use it in the experimental part of this work. Alternatively, one may choose bounds d_i to make use of as many qubits as the underlying quantum architecture offers with a hope that short enough vectors still get encoded in the search space. There also exists a more in-depth approach that determines bounds on $|x_i|$ based on the reduction of the corresponding dual basis [9].

III. FIGURES OF MERIT FOR RESOURCE ESTIMATION

In evaluating the implementations we give, we benchmark them with respect to the following figures of merit.

1) *Space complexity—the number of qubits that are needed:* This can be further divided into "input" qubits and ancillary that are all initialized to $|0\rangle$.
2) *Time complexity:* The depth (delay) of the circuit/algorithm is the largest number of 1×1 and 2×2 quantum gates applied sequentially to a qubit.
3) *Quantum cost—number of 1×1 and 2×2 reversible gates required to implement a given circuit after a decomposition from higher qubit gates (3 and more) is performed:* This is a standard metric and has been used, e.g., in [29].
4) *Number of T-gates:* Such gates are more expensive when implemented in a fault-tolerant way because the most commonly used error-correcting codes do not support transversal T-gates and instead rely on magic state distillation.

IV. ORACLE CIRCUIT DESIGN

To construct a Grover's oracle for the SVP, we will first recall how to turn a search problem to a Grover's oracle, and then, we will outline the steps required for our special case, the SVP. The first thing to do is to define a Boolean function that returns 1 for inputs that are solutions to the search and 0 in all other cases. Then, the corresponding function is expressed as a Boolean circuit and then as a reversible circuit, which in turn is made to a unitary circuit that acts as the oracle $O_f|x, y >= |x, y \oplus f(x) >$ that uses one extra output register.

Coming back to the SVP, we can break the oracle construction in the following steps.

Step 1—Define the search space: Given a basis, one defines the search space as all the lattice vectors that can be produced as linear combinations of the basis vectors, where each of the coefficients x_i of the basis vectors are integers whose absolute values are bounded by a constant d_i (see Section II-F). If this constant is chosen suitably, the shortest vector of the full lattice is within the vectors spanned in the resulting subspace with a high probability. Joseph et al. [8] and Albrecht et al. [9] demonstrate that d_i scales as $O(\log n)$ for an n-dimensional basis. This step gives us the cardinality of the search space (used in Grover's algorithm), which we denote by N.

A result from [9] states that if a lattice vector has length less than A, so that $\|x_1\mathbf{b}_1 + \cdots + x_n\mathbf{b}_n\| \leq A$, then, on average, it is sufficient to choose $x_i \leq A \cdot \|\hat{\mathbf{b}}_i\|$, where $\hat{\mathbf{b}}_i$ is the ith basis vector of the dual basis $\hat{\mathbf{B}}$. This result translates a bound on the length of the lattice vector into one on the length of the coefficient x_i. Bounding the coefficient vectors, therefore, depends on the choice of A and the quality of the dual basis. The authors describe a preprocessing algorithm, which takes in a general lattice basis and reduces the dual and primal bases. Assuming a search space parameterized by the Gaussian heuristic, they show that an HKZ basis can be found (thus solving the SVP) with

$$Q(n) \leq \frac{3}{2}n\log(n) - 2.26n + O(\log n) \tag{3}$$

qubits. This means that the search space is $N = 2^{cn\log n+o(n)}$, where $c = \frac{3}{2}$. We note that classical enumeration is done in the same search space and, thus, would require the time complexity of the order of that number.

Step 2—Define the solution space: We do not (a priori) know the length of the shortest vector. However, it is essential to define what a solution to the search problem is before the run of a Grover's algorithm. We define a solution space to be a subset of vectors of the search space (from Step 1) that are smaller than a certain threshold T. Finding a suitable T is nontrivial as care needs to be given not to choose T overly small or overly large. In such cases, we would either miss the solution to the SVP or we would converge toward a subset spanned by a large number of lattice vectors, most of which are too large to be considered a valid solution of the SVP. We could use Minkowski's bound [28] to obtain an upper bound on the length of the shortest vector $\lambda_1(\mathcal{L}) \leq \sqrt{n} \cdot D^{1/n}$, where D is the covolume of the lattice, equivalent to the modulus of the determinant for any basis. However, being a crude overapproximation, a better approach is to use the Gaussian heuristics. The Gaussian heuristic estimates the number of lattice points inside a given ball $\mathcal{B}$ as $\mathsf{Vol}(\mathcal{B})/\mathsf{Vol}(\mathcal{L})$. It gives a good approximation to the length of the shortest vector as $\lambda_1(\mathcal{L}) \simeq gh(\mathcal{L}) = \sqrt{n/2\pi e} \cdot D^{1/n}$. Choosing $T = \mathrm{gh}(\mathcal{L}))$ thus means that the ball of radius A contains one lattice point for most lattices, which is the shortest vector. Due to the definition of the Gaussian heuristic as $\mathsf{Vol}(B)/\mathsf{Vol}(\mathcal{L})$, the expected number of solutions is 1, which is necessary to determine the number of iterations for Grover's algorithm. Alternatively, one might use a constant multiple of $\mathrm{gh}(\mathcal{L})$ with constant slightly larger than 1 to improve a probability of not missing out the shortest lattice vector. The optimal setting is outside scope of this article and is left as future work.

Step 3—Find the Boolean a function for this search problem: This step breaks into two parts. The first part is to evaluate the length of a given vector. Observe that the following equation calculates a resulting squared length of a lattice vector given a row-wise lattice basis matrix $\mathbf{B}$ (of dimension $n \times m$) and a coefficient vector $x = (x_0, \ldots, x_{n-1})$

$$\sum_{j=0}^{m-1}\left[\sum_{i=0}^{n-1} x_i \mathbf{B}_{ij}\right]^2 .$$

As we vary the coefficients x_i within the bounds determined in Step 1, the output register takes on values of squared lengths of all possible lattice vectors inside our predefined search space. It is important to note that computing the length is independent of the threshold length for which we consider a vector to be a solution (i.e., on our choices in Step 2); therefore, the resources required for a single oracle call do not depend on the choices of multiplicative constant for the Gaussian heuristic.

Second, we perform a projection into two outcomes, separating solutions (below the desired length) from the nonsolutions (larger vectors). Again, for this projection, the actual length defining the solution does not change the oracle's resource cost, which is not the case for the overall resource cost as it influences Grover iteration count, as discussed further in Section V-C.

Given an $n \times m$ basis matrix $\mathbf{B}$, the desired quantum oracle implements a function $f : (x_0, \ldots, x_{n-1}) \rightarrow \{0, 1\}$ defined as

$$(x_0, \ldots, x_{n-1}) \rightarrow \begin{cases} 1, & \text{if } \sum_{j=0}^{m-1}\left[\sum_{i=0}^{n-1} x_i \mathbf{B}_{ij}\right]^2 \leq T^2 \\ 0, & \text{otherwise} \end{cases} \quad (4)$$

where $(x_0, \ldots, x_{n-1}) \in \mathbb{Z}^n$ are bounded coefficients over which the enumeration is being performed (as determined in Step 1) and T is ideally a tight upper bound on the length of the shortest lattice vector (as determined in Step 2).

Step 4—Construction of a quantum circuit implementing the oracle: Given that we know the analytical formula for $f(x)$ found in the previous step, we can now construct a reversible unitary quantum circuit for O_f that acts on the Hilbert space of the input qudits $|x|$, where each of $x_i \in \{x_0, \ldots, x_{n-1}\}$ is a $2d_i$-qudit system, as explained in Section II-F, and a single output qubit $|y>$, which gets flipped if and only if $|x>$ refers to a lattice vector belonging to a solution space. See Section II-D for a more detailed explanation on the action of the oracle and Section IV-A for a further discussion about how to implement the qudits $|x>$ in the given ranges. One can either use standard results from literature about techniques to implement basic arithmetic operations using quantum circuits or one can use a quantum circuit compiler translating an analytical formula into a gate-based quantum circuit. The first option allows for a more flexible tradeoff in between different kinds of quantum resources (number of additional ancillary qubits, circuit depth, quantum cost, and gate set), essentially tailoring the oracle precisely to a specific underlying quantum architecture. In such a case, one would break (4) into steps of elementary arithmetic operations (as shown in the Appendix) and consider the optimal way for their implementation. We list some of the state-of-the-art techniques to implement the arithmetic operations needed to build the oracle included in a gate-based quantum circuit model: for addition [30], [31], for multiplication [32], [33], [34], and for subtraction [35]. The final comparison of the squared length of the vector with a threshold T^2 (determined in Step 2) is performed as a subtraction followed by a CNOT gate controlled by a subtraction overflow qubit (indicating that the subtraction result is below zero) and a target qubit $|y>$, which is the output qubit, as explained in Section II-D. Alternatively, one might use a quantum circuit compiler that takes as input an analytical equation of the function and outputs a quantum circuit that implements it. Many such software tools exist, e.g., [36], [37], [38], [39]. Since the oracle will be reused multiple number of times during the execution of Grover's algorithm, it is essential that it uncomputes its action on the input qudit register $|x>$ so that the same register can be used as input to the subsequent runs of the oracle. This can be trivially achieved by applying an inverse gate for each of the gates in a reverse order.

Step 5—Construction of the Grover's search circuit: Once Grover's reflection $H^{\otimes n}\hat{M}_0 H^{\otimes n}$ is appended to the oracle O_f, a circuit for a single Grover's iteration $\hat{G}$ (see Fig. 2) is constructed. As shown in Fig. 1, in order to implement the full Grover's search algorithm for the SVP, this Grover's iteration needs to be repeated a number of times. This number depends on the ratio of the cardinality of the search space N and the cardinality of the solution space M, and it has been determined in [11] that exactly $\lceil \frac{\pi}{4} \rceil \sqrt{N/M}$ iterations are needed. The resulting circuit (prepended by $H^{\otimes n}$ as in Fig. 1) implements Grover's algorithm for the SVP. It can then by compiled using various software techniques and decomposed into an actual gate set supported by an underlying quantum architecture.

A. FURTHER ON STEP 3: ENCODING THE BOUNDED INTEGER ENUMERATION COEFFICIENTS IN TERMS OF QUBITS

We encode each coefficient x_i in a binary representation, as shown in [8] and [9]. Recall that in Step 1, we chose a bound $|x_i| \leq d_i$ for each of the coefficients. Given $n \times n$ row-wise lattice basis matrix $\mathbf{B}$, the enumeration finds an optimal linear combination of the rows of $\mathbf{B}$ to form the shortest lattice vector. Given the enumeration coefficients $(x_0, \ldots, x_{n-1})$, we express them in a binary expansion

$$x_i = -d_i + \sum_{j=0}^{\lfloor \log 2d_i \rfloor} c_{ij} 2^j \tag{5}$$

where each c_{ij} is a newly introduced binary variable. Note that Albrecht et al. [9] propose different encodings of integer coefficients as a technique to avoid the trivial solution of the SVP (i.e., the zero vector solution) as

$$x_i = -d_i + \zeta_i d_i + \omega_i(d_i + 1) + \sum_{j=0}^{\lfloor \log(a-1)d_i \rfloor - 1} c_{ij} 2^j$$

where the enumeration coefficients are $\zeta_i, \omega_i, c_{i0}, c_{i1}, \ldots$ with the advantage that a penalty $L \prod_i \zeta_i$, where a penalty value $L >> 0$ can be introduced more efficiently as if (5) is used (in such a case, the penalty becomes $L \prod_{i,j} c_{ij}$, which is more costly to implement). This different encoding, however, requires more qubits. In the approach using variational algorithms [7], [8], [9], [10], [19], one needs to ensure that the trivial solution does not correspond to the minimal eigenvalue of the corresponding problem Hamiltonian operator, since one performs a minimization and not excluding the zero vector would result in the approach failing to identify the shortest vector by returning the zero vector. In search algorithms, before the final measurement, one obtains a vector that is (with high probability) in the solution subspace, and specifically in an equal superposition of the different classical solutions. In our approach to the SVP, one defines the solution space as an intersection of the lattice points with a ball placed in the origin with a radius T (determined in Step 2). Therefore, if the ball radius T is optimal, the solution space, even though still containing the trivial solution (and hence increasing the cardinality of solution space by one), also contains sufficiently short lattice vectors (within the ball radius). Generating an equal superposition in a subspace that contains solutions and the zero vector is not a problem, since each solution and the zero vector have equal probability, and by simply repeating the process, we can make the probability of obtaining the zero vector negligible. For example, even with a single nonzero solution in the solution space, to obtain one solution (on average), it suffices to repeat twice.

Note that the alternative of avoiding the zero vector by an explicit construction of the Grover's oracle would require significant extra resources and would thus be redundant. To summarize, after a very few runs of the Grover's algorithm, we sample a nonzero short lattice vector within a bound T with a very high probability.

V. RESOURCE ESTIMATION

In this section, we analyze both the circuit for the oracle O_f and the circuit for the full Grover's search algorithm. Section V-A provides a theoretical analysis to prove scalings for resources required to implement the single oracle O_f and finds the polynomial scaling of space complexity (quadratic) and the quantum cost (quintic) in dimensions of the lattice basis. We also prove that time complexity scales linearly. Section V-B then presents experimental results after the circuit for the oracle O_f has been compiled by Quipper [37] and plots the experimentally determined scalings of resource requirements. We find that the experimentally determined asymptotes agree with the theoretical analysis, and we provide exact numbers for the quantum resources if the circuit is decomposed into the Clifford+T gate set. Finally, in Section V-C, using the costs for the oracle, we obtain the cost of a single Grover's iteration and, subsequently, the overall cost of Grover's search algorithm by repeating the iteration sufficient number of times.

A. ANALYTICAL ANALYSIS OF ORACLE RESOURCE REQUIREMENTS

In this section, we present a high-level circuit design. Let $\mathbf{B}$ be an $n \times m$ row-wise matrix of the lattice basis such that the first column is nonnegative (this can be achieved by the multiplication of some of the rows of $\mathbf{B}$ by -1 as needed). This saves a few quantum operations as the sign of the result of multiplication $x_i \mathbf{B}_{i1}$ depends solely on the sign of x_i (see the Appendix for a more detailed discussion). Recall that formula [(4), which will denote γ] to be implemented as the quantum circuit is a function $\gamma := f : (x_0, \ldots, x_{n-1}) \to \{0, 1\}$ defined as

$$(x_0, \ldots, x_{n-1}) \to \begin{cases} 1, & \text{if } \sum_{j=0}^{m-1} \left[\sum_{i=0}^{n-1} x_i \mathbf{B}_{ij} \right]^2 \leq T^2 \\ 0, & \text{otherwise.} \end{cases}$$

We proceed to analyze the scalings of

1) $S(\gamma)$, a function that returns the space complexity of implementing a quantum circuit corresponding to the function γ;
2) $T(\gamma)$, a function that returns the time complexity of a quantum circuit implementing the function γ;
3) $C(\gamma)$, a function that returns quantum cost of a quantum circuit implementing the function γ.

These three functions express the figures of merit that we want to consider and are essentially functions/properties of quantum circuits. We will, therefore, break these calculations to the steps/subcircuits that give this function. For notational simplicity, we give as arguments of these functions the function/operations that the corresponding circuit implements.

Suppose that we use the qudit encoding from (5), which is a standard, and probably the optimal, choice, as discussed in Section IV-A. We split the calculation of γ into subsequent steps, which are also depicted in Fig. 5. Note that, to the best of our knowledge, all the existing proposals for quantum arithmetic circuits acting on two operands assume that both the operands are of the same size. Suppose, for example, that we want to implement a sum of two integer quantities K and L encoded in k and l number of qubits, respectively, with $k \leq l$. Then, the first operand needs to be padded with $l-k$ extra qubits initialized to zero to make the encoding of both integer quantities of equal length. Therefore, in this case, $S(K+L) = 2\max(k,l)$.

1) *Encoding the enumeration coefficients:* By (5), $x_i = \sum_{k=0}^{\lfloor \log 2d_i \rfloor} 2^k c_k - d_i$, with each c_k being a binary variable (taking values 0 or 1). We implement each c_k with a single logical qubit. Since, in the calculation of x_i, a subtraction operation is needed, the number of qubits needed to implement each enumeration coefficient is

$$\begin{aligned}\mathrm{S}(x_i) &= 2\max(\lfloor \log 2d_i \rfloor + 1, \lfloor \log d_i \rfloor + 1) + o(\log d_i)\\ &= 2\lfloor \log 2d_i \rfloor + 1 + o(\log d_i)\\ &= \Theta(\log d_i)\end{aligned}$$

where $o(\log d_i)$ accounts for an additional qubit representing a sign of subtraction and additional ancillary qubits depending on a specific subtraction implementation. Because a quantum implementation of subtraction has both linear time complexity and linear quantum cost in the number of input qubits [35]

$$T(x_i) = \Theta(S(x_i)) = \Theta(\log d_i)$$
$$C(x_i) = \Theta(S(x_i)) = \Theta(\log d_i).$$

2) *Multiplication of an enumeration coefficient with an element of a basis vector:* Once an enumeration coefficient x_i is calculated inside the circuit, the next step is to multiply it with an element of a basis vector. Even though x_i needs to be multiplied with each element $\mathbf{B}_{ij}$ for all $0 \leq j \leq n-1$, in this step, we consider a multiplication for a single value of j. An encoding of $\mathbf{B}_{ij}$ requires an additional qubit for $j \geq 1$ (as we assumed $\forall i: \mathbf{B}_{i0} \geq 0$). The space complexity is therefore

$$\begin{aligned}\mathrm{S}(x_i\mathbf{B}_{ij}) &= 2\max(Q(x_i), (\lfloor \log \mathbf{B}_{ij} \rfloor + 1) + 1_{\text{if } j\geq 1})\\ &= 2\max(\theta(\log d_i), \theta(\log \mathbf{B}_{ij}))\\ &= \Theta(\log \max(d_i, \mathbf{B}_{ij})).\end{aligned}$$

There are many options to choose how to implement the multiplication operation with different asymptotics. Nie et al. [34] provide a comparison between state-of-the-art approaches. We continue our analysis with a choice of Schönhage–Strassen algorithm [34] as it has, in our opinion, the best ratio of space complexity to quantum cost (being $O(\log_2^2 n)$ and $O(n \cdot \log n \cdot \log\log n)$, respectively), while the time complexity is relatively small compared to most of other approaches with polynomial time complexities

$$\begin{aligned}\mathrm{T}(x_i\mathbf{B}_{ij}) &= \Theta(\log^2(\mathrm{S}(x_i\mathbf{B}_{ij}))) + \mathrm{T}(x_i)\\ &= \Theta(\log^2 \log^2 \max(d_i, \mathbf{B}_{ij})) + \Theta(\log d_i)\\ &= \Theta(\log d_i)\end{aligned}$$

$$\begin{aligned}\mathrm{C}(x_i\mathbf{B}_{ij}) &= \Theta(\mathrm{S}(x_i\mathbf{B}_{ij}) \times \log \mathrm{S}(x_i\mathbf{B}_{ij})\\ &\quad \times \log\log \mathrm{S}(x_i\mathbf{B}_{ij})) + \mathrm{C}(x_i)\\ &= \Theta(\log \max(d_i, \mathbf{B}_{ij})\\ &\quad \times \log\log \max(d_i, \mathbf{B}_{ij})\\ &\quad \times \log\log\log \max(d_i, \mathbf{B}_{ij})) + \Theta(\log d_i)\\ &= \Theta(\log \max(d_i, \mathbf{B}_{ij})\\ &\quad \times \log\log \max(d_i, \mathbf{B}_{ij})\\ &\quad \times \log\log\log \max(d_i, \mathbf{B}_{ij})).\end{aligned}$$

3) *Multiplication of an enumeration coefficient with a jth column of* $\mathbf{B}$*:* We proceed to calculate resource requirements for the inner sum of γ, the expression $\sum_{i=0}^{n-1} x_i\mathbf{B}_{ij}$. To implement an addition operation taking n operands, the least costly approach is to stack $\Theta(n)$ circuits implementing addition for two operands in serial–parallel structure resembling a tree with $\Theta(\log n)$ layers. Each layer reduces the number of operands to be summed up by a half (see Fig. 5 for a drawing). No other qubits need to be introduced (except possibly some additional ancillary qubits which scale at worst as the scalings of the operands). Let $\kappa_j := \max(\max_i d_i, \max_i \mathbf{B}_{ij})$ be the length of the largest input register out of $|d_0\rangle, |d_1\rangle, \ldots, |d_{n-1}\rangle, |B_{0j}\rangle, |B_{1j}\rangle, \ldots, |B_{n-1,j}\rangle$. Because a quantum implementation of addition has both linear time complexity and linear quantum cost in the size of operands [35], it follow that

$$\begin{aligned}\mathrm{S}\left(\sum_{i=0}^{n-1} x_i\mathbf{B}_{ij}\right) &= n \max_i \mathrm{S}(x_i\mathbf{B}_{ij})\\ &= \Theta(n \log \max(\max_i d_i, \max_i \mathbf{B}_{ij}))\end{aligned}$$

$$= \Theta(n \log \kappa_j)$$

$$\begin{aligned}
\mathrm{T}\left(\sum_{i=0}^{n-1} x_i \mathbf{B}_{ij}\right) &= \Theta(\log n)\Theta(\mathrm{S}(x_i \mathbf{B}_{ij})) + \mathrm{T}(x_i \mathbf{B}_{ij}) \\
&= \Theta(\log n \times \log \kappa_j) + \Theta(\log \kappa_j) \\
&= \Theta(\log n \times \log \kappa_j)
\end{aligned}$$

$$\begin{aligned}
\mathrm{C}\left(\sum_{i=0}^{n-1} x_i \mathbf{B}_{ij}\right) &= \Theta(n)\Theta(\mathrm{S}(c_i \mathbf{B}_{ij})) \\
&\quad + \Theta(n \times \mathrm{C}(x_i \mathbf{B}_{ij})) \\
&= \Theta(n \times \log \kappa_j) \\
&\quad + \Theta(n \times \log \kappa_j \times \log\log \kappa_j \\
&\quad \times \log\log\log \kappa_j) \\
&= \Theta(n \times \log \kappa_j \times \log\log \kappa_j \\
&\quad \times \log\log\log \kappa_j).
\end{aligned}$$

4) *Taking square of a resulting vector elements:* To calculate a square of a register $|\psi\rangle$, one can either initialize a new quantum register with the same length or copy the values of logical qubits from $|b\rangle$ (b being a bitstring), essentially creating $|b\rangle|b\rangle$. Note that the copy operation does not conflict with the no-cloning theorem because $|\psi\rangle$ is a register of qubits having values in the computational basis (0 or 1). One can then use a circuit for multiplication to calculate $|b^2\rangle$. Alternatively, one can use a circuit proposed in [40] that presents an inverted circuit for the square root operation based on the nonrestoring digit recurrence method [41], which, in practice, lowers time complexity and quantum cost, although the asymptotes stay the same. We will ignore the cost for the copy operation in the analysis: 1) it would not change the asymptotic scalings and 2) if the latter approach is followed, the copy operation is not performed. So, we find that

$$\begin{aligned}
\mathrm{S}\left(\left[\sum_{i=0}^{n-1} x_i \mathbf{B}_{ij}\right]^2\right) &= 2\mathrm{S}\left(\sum_{i=0}^{n-1} x_i \mathbf{B}_{ij}\right) \\
&= \Theta(n \log \kappa_j)
\end{aligned}$$

$$\begin{aligned}
\mathrm{T}\left(\left[\sum_{i=0}^{n-1} x_i \mathbf{B}_{ij}\right]^2\right) &= \Theta\left(\log^2 \mathrm{S}\sum_{i=0}^{n-1} x_i \mathbf{B}_{ij}\right) \\
&\quad + T\left(\sum_{i=0}^{n-1} x_i \mathbf{B}_{ij}\right) \\
&= \Theta\left(\log^2(n \log \kappa_j)\right) \\
&\quad + \Theta(\log n \times \log \kappa_j) \\
&= \Theta(\log^2(n \log \kappa_j))
\end{aligned}$$

$$\begin{aligned}
\mathrm{C}\left(\left[\sum_{i=0}^{n-1} x_i \mathbf{B}_{ij}\right]^2\right) &= \Theta\left(\mathrm{S}\left(\sum_{i=0}^{n-1} x_i \mathbf{B}_{ij}\right)\right. \\
&\quad \times \log \mathrm{S}\left(\sum_{i=0}^{n-1} x_i \mathbf{B}_{ij}\right) \\
&\quad \left.\times \log\log \mathrm{S}\left(\sum_{i=0}^{n-1} x_i \mathbf{B}_{ij}\right)\right) \\
&\quad + \Theta(n)\mathrm{C}\left(\sum_{i=0}^{n-1} x_i \mathbf{B}_{ij}\right) \\
&= \Theta((n \log \kappa_j) \times \log(n \log \kappa_j) \\
&\quad \times \log\log(n \log \kappa_j)) \\
&\quad + \Theta(n)\Theta\left(n \times \log \kappa_j\right. \\
&\quad \left.\times \log\log \kappa_j \times \log\log\log \kappa_j\right) \\
&= \Theta\left(n^2 \times \log \kappa_j \times \log\log \kappa_j\right. \\
&\quad \left.\times \log\log\log \kappa_j\right).
\end{aligned}$$

5) *Calculating a squared length of the resulting vector by performing the outer sum:* The last step to calculate a squared length of the lattice vector corresponding to the enumeration coefficients $x_0, \ldots, x_{n-1}$ is to take the outer sum, i.e., to implement $\sum_{j=0}^{m-1}[\sum_{i=0}^{n-1} x_i \mathbf{B}_{ij}]^2$. We follow the strategy, as outlined in Step 3, to implement an addition of m operands. Let $\hat{\kappa} := \max_j \kappa_j$, i.e., it is the length of the largest input register out of $|d_0\rangle, |d_1\rangle, \ldots, |d_{n-1}\rangle, |B_{00}\rangle, \ldots, |B_{0j-1}\rangle, |B_{1,0}\rangle, \ldots, |B_{n-1,j-1}\rangle$. The costs of required resources are then

$$\begin{aligned}
\mathrm{S}\left(\sum_{j=0}^{m-1}\left[\sum_{i=0}^{n-1} x_i \mathbf{B}_{ij}\right]^2\right) &= m \max_{i,j} \mathrm{S}\left(\left[\sum_{i=0}^{n-1} x_i \mathbf{B}_{ij}\right]^2\right) \\
&= m \max_j \Theta(n \log \kappa_j) \\
&= \Theta(mn \log \hat{\kappa})
\end{aligned}$$

$$\begin{aligned}
\mathrm{T}\left(\sum_{j=0}^{m-1}\left[\sum_{i=0}^{n-1} x_i \mathbf{B}_{ij}\right]^2\right) &= \Theta(\log m) \\
&\quad \times \Theta\left(\mathrm{S}\left(\left[\sum_{i=0}^{n-1} x_i \mathbf{B}_{ij}\right]^2\right)\right) \\
&\quad + \mathrm{T}\left(\left[\sum_{i=0}^{n-1} x_i \mathbf{B}_{ij}\right]^2\right) \\
&= \Theta(\log m)\Theta\left(n \log \hat{\kappa}\right) \\
&\quad + \Theta(\log^2(n \log \hat{\kappa})) \\
&= \Theta\left(n \log m \times \log \hat{\kappa}\right)
\end{aligned}$$

$$\begin{aligned}
\mathrm{C}\left(\sum_{j=0}^{m-1}\left[\sum_{i=0}^{n-1}x_i\mathbf{B}_{ij}\right]^2\right) &= \Theta(m)\\
&\quad\times\Theta\left(\mathrm{S}\left(\left[\sum_{i=0}^{n-1}x_i\mathbf{B}_{ij}\right]^2\right)\right)\\
&\quad+\Theta\left(m\mathrm{C}\left(\left[\sum_{i=0}^{n-1}x_i\mathbf{B}_{ij}\right]^2\right)\right)\\
&=\Theta(m)\Theta\left(n\log\hat{\kappa}\right)\\
&\quad+\Theta\left(m\times n^2\times\log\hat{\kappa}\right.\\
&\quad\left.\times\log\log\hat{\kappa}\times\log\log\log\hat{\kappa}\right)\\
&=\Theta(m\times n^2\times\log\hat{\kappa}\\
&\quad\times\log\log\hat{\kappa}\times\log\log\log\hat{\kappa}).
\end{aligned}$$

6) *Deciding whether the resulting vector lies in a solution space:* The last step is to implement the comparison $\sum_{j=0}^{m-1}[\sum_{i=0}^{n-1}x_i\mathbf{B}_{ij}]^2 \leq T^2$. For this, we use a subtractor to calculate $\sum_{j=0}^{m-1}[\sum_{i=0}^{n-1}x_i\mathbf{B}_{ij}]^2 - T^2$, and we check if the result overflows through zero. Similarly as in Step 1, we need to have the same number of qubit inputs for both operands. Since it is expected that we choose T^2 to be lower than the largest vector from our search space, we know that $\mathrm{S}(\sum_{j=0}^{m-1}[\sum_{i=0}^{n-1}x_i\mathbf{B}_{ij}]^2) \geq \mathrm{S}(T^2)$ and hence

$$\begin{aligned}
\mathrm{S}\left(\sum_{j=0}^{m-1}\left[\sum_{i=0}^{n-1}x_i\mathbf{B}_{ij}\right]^2\leq T^2\right)
&=2\mathrm{S}\left(\sum_{j=0}^{m-1}\left[\sum_{i=0}^{n-1}x_i\mathbf{B}_{ij}\right]^2\right)\\
&=\Theta\left(mn\log\hat{\kappa}\right).
\end{aligned}$$

We again use a circuit for subtraction as in Step 1, and therefore, we have a linear scaling of time complexity and quantum cost in the number of input qubits

$$\begin{aligned}
\mathrm{T}\left(\sum_{j=0}^{m-1}\left[\sum_{i=0}^{n-1}x_i\mathbf{B}_{ij}\right]^2\leq T^2\right)
&=\Theta\left(\mathrm{S}\left(\sum_{j=0}^{m-1}\left[\sum_{i=0}^{n-1}x_i\mathbf{B}_{ij}\right]^2\right)\right)\\
&\quad+\mathrm{T}\left(\sum_{j=0}^{m-1}\left[\sum_{i=0}^{n-1}x_i\mathbf{B}_{ij}\right]\right)\\
&=\Theta(mn\log\hat{\kappa})+\Theta\left(n\times\log m\times\log\hat{\kappa}\right)\\
&=\Theta(n\times m\times\log\hat{\kappa})
\end{aligned}$$

$$\begin{aligned}
\mathrm{C}\left(\sum_{j=0}^{m-1}\left[\sum_{i=0}^{n-1}x_i\mathbf{B}_{ij}\right]^2\leq T^2\right)
&=\Theta\left(\mathrm{S}\left(\sum_{j=0}^{m-1}\left[\sum_{i=0}^{n-1}x_i\mathbf{B}_{ij}\right]^2\right)\right)\\
&\quad+\mathrm{C}\left(\sum_{j=0}^{m-1}\left[\sum_{i=0}^{n-1}x_i\mathbf{B}_{ij}\right]^2\right)\\
&=\Theta(mn\log\hat{\kappa})+\Theta\left(m\times n^2\times\log\hat{\kappa}\right.\\
&\quad\left.\times\log\log\hat{\kappa}\times\log\log\log\hat{\kappa}\right)\\
&=\Theta\left(m\times n^2\times\log\hat{\kappa}\right.\\
&\quad\left.\times\log\log\hat{\kappa}\times\log\log\log\hat{\kappa}\right).
\end{aligned}$$

Letting $n = m$ (i.e., for the case of square lattices) and taking $\hat{\kappa} = \Theta(\log n)$ (since it grows with the number of qubits dedicated to each enumeration coefficient), we conclude that using the choices for the circuits implementing the arithmetic operations. The scalings listed in Table 1 would be the resource scalings.

TABLE 1. Calculated Asymptotical Quantum Resource Requirements for a Single Instance of an Oracle O_f Given a Full-Rank Lattice Basis of Dimension n

Space Complexity	$\Theta(n^2\times\log\log n)$
Time Complexity	$\Theta(n^2\times\log\log n)$
Quantum Cost	$\Theta(n^3\times\log\log n\times\log\log\log n \times\log\log\log\log n)$

TABLE 2. Oracle Resource Requirements Found by Quipper With log *n* Qubits Per Each Enumeration Coefficient

Lattice Dimension	2	5	10	20
Space Complexity	29	263	1154	5165
Time Complexity	344	3058	7248	17352
Quantum Cost	2006	70996	467774	2817236
Number of T-Gates	28	4718	80858	542674
T-depth	62	736	1220	2866

Lattice Dimension	30	40	50
Space Complexity	11345	23246	36056
Time Complexity	23360	38854	45440
Quantum Cost	6311554	15910146	24831782
Number of T-Gates	1258752	3232930	5097846
T-depth	3932	6336	7640

B. EXPERIMENTAL RESOURCE REQUIREMENT SCALINGS DETERMINED BY CLASSICAL COMPILATION

We compiled the circuit for quantum oracle O_f (4) using Quipper [37], including the uncomputation of any intermediate action performed on the input register. The resource requirements are given in Table 2 and are plotted in Fig. 3. We performed the best-fit of the experimental results to find the following scalings:

$$\text{Space complexity} \approx 5.14n^2\log\log(n)$$

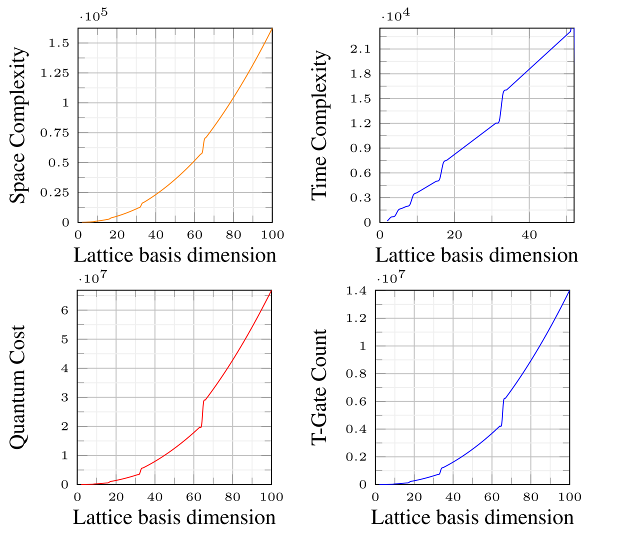

FIGURE 3. Experimental resource requirements for SVP oracle circuit as found by Quipper. This plots extended dataset of the data found in Table 2. The "sudden" jumps at dimensions being power of 2 (*x*-axis being.., 32,64,..) are due to rounding in determining the number of qubits per enumeration coefficient, which is ⌈log *n*⌉.

$$\begin{aligned}&+1101.12n\log\log(n)\\&+3348.47\log\log(n)-2908.46n\\&+4682.92\\ \text{Time complexity}\approx{}&0.12n^2\log\log(n)-749.05\log\log(n)\\&+494.23n-455.1\\ \text{Quantum cost}\approx{}&10.92n^3l(n)+181183.18nl(n)\\&-6442137.89l(n)-246992.36n\\&-39111224.06\\ \text{T-gate count}\approx{}&2.34n^3l(n)+38801.38nl(n)\\&-1380667.01l(n)-52831.25n\\&-837651.82\end{aligned}$$

where

$$\begin{aligned}l(n):&=\log\log(n)\log\log\log\\&\times(n)\log\log\log\log(n).\end{aligned}$$

To find the scaling, *curve_fit* function from Python's library *scipy* has been used to fit a function of the form

$$\begin{aligned}f(n)&=an^2\log\log(n)+bn^2+cn\log\log(n)\\&+d\log\log(n)+en+f\end{aligned}$$

with coefficients $a, b, \ldots, f$ to the calculated space complexity, time complexity, and T-depth requirements and a function of the form

$$\begin{aligned}f'(n)&=an^3l(n)+bn^2l(n)+cnl(n)+dl(n)+en^3\\&+fn^2+gn+h\end{aligned}$$

with coefficients $a, b, \ldots, h$ to the calculated quantum cost and T-gate count requirements. The data that have been fitted have been calculated for all the lattice dimensions between 2 and 100. A subset of the data is presented in Table 2.

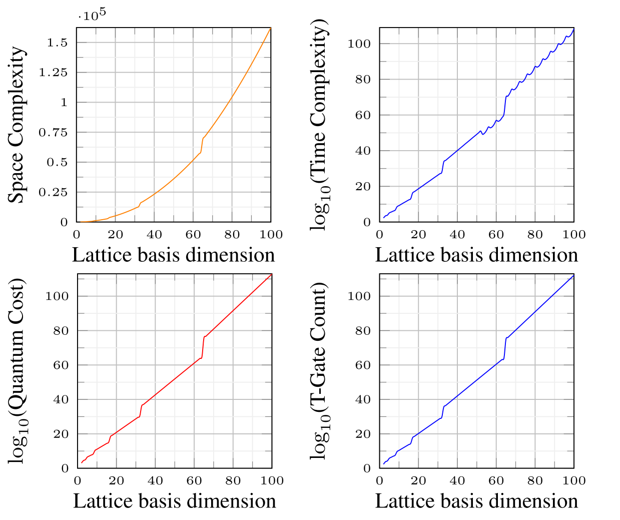

FIGURE 4. Experimental resource requirements for Grover's search SVP routine as found by Quipper. This plots extended dataset of the data found in Table 4. Similarly as in Fig. 3, the "sudden" jumps are caused by rounding a function of the number of qubits to the nearest higher integer. Note that the space complexity remains the same for a single oracle and Grover's search algorithm: once an oracle is constructed, no additional logical qubits are needed to run the full Grover's search. The graph is plotted again due to reference only.

C. SOLVING SVP USING GROVER'S ALGORITHM AND ITS RESOURCES

Given the oracle implementation of the previous section, we can use Section II and compute the resources required to find solutions of the SVP for different dimensions. This is done by combining the oracle with other gates to form the Grover iteration and then repeating it a sufficient number of times, as described in Section II-D. In Fig. 4, we give the breakdown of the resources costs for those. Table 4 extracts some of the data from Fig. 4 and provides the exact calculated results for several chosen lattice dimensions. When run as part of Grover, both uncomputation (which doubles the depth) and repeated execution $\lceil\frac{\pi}{4}\rceil\times\sqrt{N/M}$ times are required, with N being the size of the search space as determined in Step 1 of Section IV and scales as $2^{\frac{3}{2}n\log n+o(n)}$ and M being the size of the solution space as determined in Step 2 of Section IV. Given the integer bounds on the enumeration coefficients $|x_i|\leq d_i$, let $|\text{bin}(d_i)|$ be a minimal number of qubits needed to encode the coefficient in the given range $[-d_i, d_i]$. Then, $N=\sum_i|\text{bin}(d_i)|$, and this quantity scales as $O(n\log n)$ as $0\leq i\leq n$, and on average, it is enough to assign $\log n$ qubits per coefficient [9]. If the bound T in Step 2 of Section IV was chosen optimally, the search space would contain a zero vector, which is always a trivial solution, at least two shortest lattice vectors (as each ball of radius T contains a vectors in pairs which are multiple of -1 with each other) and few extra lattice vectors. Hence, if T is chosen optimally, the size of the solution space $M>\approx 3$ will not be much larger than 3. We, therefore, use $M=3$ in the calculation of resource estimates, while we are aware that this quantity is dependent on the lattice, and accordingly, the calculated estimates might differ, because we note that the difference will be minor

TABLE 3. Estimates on Quantum Resources Required to Solve 186-D and 400-D Lattices

Lattice Dimension	186	400
Space Complexity	5.89×10^5	2.78×10^6
Time Complexity	1.25×10^{33}	5.46×10^{65}
Quantum Cost	2.33×10^{36}	7.19×10^{69}
T-Gate Count	8.36×10^{35}	1.31×10^{69}

and thus will not invalidate our estimates. To illustrate this point, consider that given a choice of the threshold T, the size of the solution space is M' that is bigger than 3 (the value assumed in our calculation of the quantum resource estimates). Consequently, the size of the solution space increases, the size of the (effective) search space decreases, and the time complexity is improved by a factor $O(1/\sqrt{M'})$. On the other hand, the increase of the size of the solution space means that it consists of more vectors that are not the actual solution to the SVP. To overcome this, $O(M')$ more runs of Grover's algorithm are required to sample the actual SVP solution. Combining these two factors, increasing M' by a nonoptimal choice of T increases the time complexity by a factor $O(1/\sqrt{M'})O(M') = O(\sqrt{M'})$. Note, however, that M' is expected to be a small constant (as the Gaussian heuristic in average provides accurate estimates), and therefore, it has a little effect on our estimates of time complexity, quantum cost, and number of T-gates.

1) EXTRAPOLATION TO "INTERESTING" LATTICE INSTANCES

Classically, the current record of the highest lattice dimension solved as marked by SVP Challenge[2] is 186. As discussed in Section II-E, to pose a threat to postquantum lattice security, one would need to solve SVPs in lattices of dimensions around 400. Using our extrapolated results from Section V-B, we estimate the required quantum resources for 186-D and 400-D lattices (see Table 3).

VI. GROVER'S IMPROVEMENT OVER BKZ ALGORITHM

We come back to the BKZ algorithm now (see Section II-E for an overview). Recall that it takes as an input a block size β and reduces a lattice basis by performing an iterative search for shortest vectors in the projected lattices of rank $\leq \beta$. As mentioned in the preliminaries, there are two main choices for β depending on what the lattice dimension is. If it is of a smaller or medium size, one can set $\beta \leq 40$ and terminate in a reasonable time. Otherwise, if the lattice dimension is large, $\beta \geq 40$ is usually chosen and the execution of the algorithm is prematurely terminated as it would not be expected to finish in a reasonable time. Further developments, described in BKZ2.0 which incorporate recent algorithmic improvements, achieve block sizes up to 120 [22]. For high values of β, the runtime of BKZ is dominated by the runtime of the SVP oracle. In a single tour, $n-1$ calls to the SVP oracle are made, and many tours may be required, although sometimes the process is halted early as this often produces

[2]https://www.latticechallenge.org/svp-challenge

TABLE 4. Grover Resource Requirements as Found by Quipper With $\log n$ Qubits Per Each Enumeration Coefficient Calculated by Including Resource Requirements for Grover's Reflection to Construct a Grover's Iteration (See Fig. 2, Section II-D) and Multiplying It By the Number of Calculated Iterations $\lceil \frac{\pi}{4}\sqrt{N/M} \rceil$

Lattice dimension	2	5	10
Number of iterations	1	83	4.75×10^5
Space complexity	29	2.63×10^2	1.15×10^3
Time complexity	1.72×10^2	1.26×10^5	1.72×10^9
Quantum Cost	1.00×10^3	2.94×10^6	1.11×10^{11}
Number of T-Gates	2.1×10^2	6.22×10^5	2.36×10^{10}
Lattice dimension	**20**	**30**	**50**
Number of iterations	5.10×10^{14}	1.71×10^{22}	2.56×10^{73}
Space complexity	5.16×10^3	1.13×10^4	3.6×10^4
Time complexity	4.42×10^{18}	2.00×10^{26}	6.15×10^{74}
Quantum Cost	7.19×10^{20}	5.40×10^{28}	8.40×10^{80}
Number of T-Gates	1.53×10^{20}	1.15×10^{28}	1.71×10^{51}
Lattice dimension	**70**	**90**	**100**
Number of iterations	3.02×10^{94}	1.03×10^{105}	6.02×10^{35}
Space complexity	8.00×10^4	1.31×10^5	1.62×10^5
Time complexity	8.47×10^{95}	2.26×10^{108}	1.17×10^{40}
Quantum Cost	1.63×10^{102}	6.95×10^{112}	4.79×10^{42}
Number of T-Gates	1.79×10^{80}	3.50×10^{101}	1.48×10^{112}

a good enough length vector, as shown in [26]. We suggest two possible approaches to integrating our quantum Grover enumeration algorithm with BKZ, given a lattice basis of dimension n.

1) Block size β remains the same as in the classical [22] scenario; however, each projected block of size β would be solved with Grover's algorithm for the SVP, described here. Since Grover's algorithm offers quadratic time improvement for the enumeration, the SVP routine would terminate in a shorter time than the classical state of the art. This approach may be necessary in the earliest stages, when the very first large fault-tolerant quantum computers are available. But this technique is suboptimal, as the large constants due to Grover imply that the advantages over classical solvers do not appear until a certain "crossover point," which in the medium term is conjectured to be computations, which take weeks or months on classical hardware [42]. Thus, seeking quadratic speedup on problem sizes already accessible to us classically is unlikely to be the most fruitful approach.
2) We increase the block size β that is possible via classical enumeration. While sieving has superior time complexity classically ($2^{0.292\beta+o(\beta)}$, versus $2^{O(\beta\log\beta)}$ for enumeration), it requires exponential space. Increasing the block size accessible via enumeration while avoiding the exponential space requirements will make quantum enumeration more appealing, although, asymptotically, classical sieving scales better than quantum enumeration. As mentioned in the previous point, the greatest quantum advantages will hail from attacking the largest problem sizes possible in one

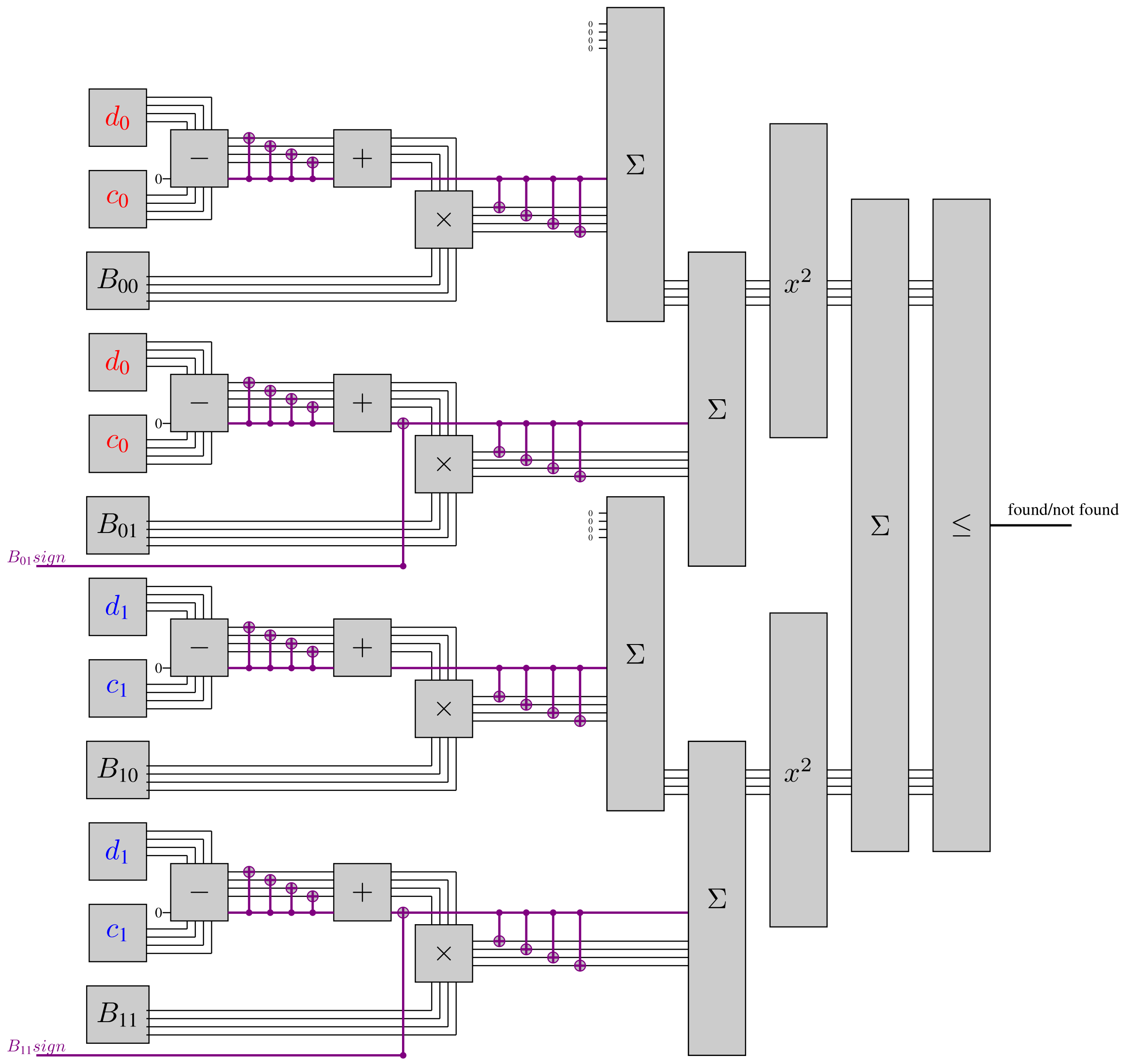

FIGURE 5. **High-level design of the SVP oracle.**

chunk; thus, maximizing block size will be the optimal approach in the long term. For example, if we consider the asymptotic Grover improvement (the factor $1/2$ in front of $\beta \log \beta$ at the exponent) and ignoring for the moment the extra overhead due to the implementation, quantumly, we could solve blocks of size $\beta \approx 70$ at the same time that classically one would solve blocks of size 40.

For Level I NIST candidates that are being standardized at the time of writing, block sizes of over 400 will be necessary to break the underlying hardness assumptions. Despite introducing Grover speedup to the problem, the time complexity of $2^{\frac{cn \log n}{2}+o(n)}$ that we prove is still asymptotically worse than classical sieving, whose main bottleneck is space requirements. We have demonstrated the impact of this Groverized enumeration circuit when integrated into BKZ, the primary approach used to attack lattice cryptosystems and hence to derive appropriate security parameters. Along with works such as [20] and [21], we also conclude that quadratic speedup quantum algorithms will not pose a significant threat to lattice-based cryptography. In order to properly compare the quantum time complexity we present here with classical algorithms, it is necessary to estimate the speed of implementation of (fault-tolerant) quantum gates, which is highly variable depending on the platform in use. Furthermore, the maturity of the technology will set limits upon the block size achievable with quantum algorithms such as ours and thus the overall attack complexity.

VII. CONCLUSION

Grover's search algorithm is one of the most important algorithms because of the generality of the speedup it offers. On the other hand, the generality arises from assuming access to an oracle that identifies the solutions, and any real use of the algorithm would require to implement such oracle without the a priori knowledge of the solution. In the case of the SVP, we overcame this difficulty by building on classical enumeration methods and prior Hamiltonian approaches to the SVP. We gave a detailed stepwise description of the oracle construction and then estimated the resources required to implement such an oracle in a quantum device. Incorporating these results to the overall quantum search algorithm of Grover led us to our main results that quantify the resources required to solve the SVP in dimension n, and we observed that the time complexity scaled as $\Theta(n^2 \log^2 n) \times 2^{\frac{3}{4}n \log n + o(n)}$ and space complexity of $\Theta(n^2 \log n)$. Our implementation and resource estimation, while interesting in its own right as an example of a quantum algorithm, does not directly have implications for cryptography—the main reason to focus on the SVP. The reason is that the scaling is still exponential, and classical state-of-the-art methods do not attempt to directly "brute-force" and search the full space. To stretch the capabilities and explore the impact that Grover's search algorithm can have for the SVP, we considered using it as a subroutine to a larger, hybrid now, algorithm. Specifically, (quantum) enumeration is the basis of the BKZ algorithm [2], where one "brute-forces" lattices of smaller dimensions and uses them to reduce the basis of a larger dimension lattice. We first demonstrated how using larger block size, possible due to better time-complexity, can improve the record for solving the SVP via the enumeration methods. At the same time, we also highlighted that for dimensions cryptographically relevant ($n \approx 400$ see NIST), our methods are nowhere close to posing a threat. Finally, to perform a fair comparison with classical methods, one needs to take into account the overheads (and delays) of quantum error correction. When speaking about concrete values, and not asymptotics, these (large) factors are crucial, but depend on many uncontrollable factors[3] and go beyond the scope of this article to compute them.

APPENDIX

GROVER'S SVP ORACLE HIGH-LEVEL CIRCUIT DESIGN

Fig. 5 shows a high-level design of a quantum circuit implementing the oracle O_f for a full-rank 2-D lattice. B_{ij} are elements of the 2×2 lattice basis matrix, d_i are bounds on the enumeration coefficients, and all c_i compose an input register to Oracle O_f. The blocks represent arithmetic operations performed, as explained step by step in Section IV. The purple lines represent additional qubits, which carry Information about sign of the binary numbers.

ACKNOWLEDGMENT

The authors thank João Doriguello for noticing an error on the time complexity evaluated in an earlier version of this article. The Haskell/Quipper code used in the experiments and the testing routines are available at https://github.com/Milos9304/svpGroversOracle.git.

[3]Error rates, speed of gates (both varying between different hardware), connectivity of qubits, quantum error correcting code, and way to perform fault-tolerant quantum operations, decoder.

REFERENCES

[1] D. Moody, *The NIST PQC Standards: Light at the End of the Tunnel*, Gaithersburg, MD, USA: Nat. Instit. Standards, 2023. Accessed: Dec. 10, 2024. [Online]. Available: https://csrc.nist.gov/events/2023/pqc-standardization-conference

[2] C.-P. Schnorr and M. Euchner, "Lattice basis reduction: Improved practical algorithms and solving subset sum problems," *Math. Program.*, vol. 66, pp. 181–199, 1994, doi: 10.1007/BF01581144.

[3] P. Shor, "Algorithms for quantum computation: Discrete logarithms and factoring," in *Proc. 35th Annu. Symp. Found. Comput. Sci.*, 1994, pp. 124–134, doi: 10.1109/SFCS.1994.365700.

[4] R. L. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," *Commun. ACM*, vol. 21, no. 2, pp. 120–126, 1978, doi: 10.1145/359340.359342.

[5] I. Blake, G. Seroussi, and N. Smart, *Elliptic Curves in Cryptography*, vol. 265. Cambridge, U.K.: Cambridge Univ. Press, 1999, doi: 10.1017/CBO9781107360211.

[6] W. Diffie and M. E. Hellman, "New directions in cryptography," *IEEE Trans. Inform. Theory*, vol. IT-22, no. 6, pp. 644–654, Nov. 1976, doi: 10.1109/TIT.1976.1055638.

[7] D. Joseph, A. Ghionis, C. Ling, and F. Mintert, "Not-so-adiabatic quantum computation for the shortest vector problem," *Phys. Rev. Res.*, vol. 2, no. 1, 2020, Art. no. 013361, doi: 10.1103/PhysRevResearch.2.013361.

[8] D. Joseph, A. Callison, C. Ling, and F. Mintert, "Two quantum ising algorithms for the shortest-vector problem," *Phys. Rev. A*, vol. 103, 2021, Art. no. 032433, doi: 10.1103/PhysRevA.103.032433.

[9] M. Albrecht, M. Prokop, Y. Shen, and P. Wallden, "Variational quantum solutions to the shortest vector problem," *Quantum*, vol. 7, Mar. 2023, Art. no. 933, doi: 10.22331/q-2023-03-02-933.

[10] Y. R. Zhu, D. Joseph, C. Ling, and F. Mintert, "Iterative quantum optimization with an adaptive problem Hamiltonian for the shortest vector problem," *Phys. Rev. A*, vol. 106, no. 2, 2022, Art. no. 022435, doi: 10.1103/PhysRevA.106.022435.

[11] L. K. Grover, "A fast quantum mechanical algorithm for database search," in *Proc. 28th Annu. ACM Symp. Theory Comput.*, 1996, pp. 212–219. doi: 10.1145/237814.237866.

[12] D. Deutsch and R. Jozsa, "Rapid solution of problems by quantum computation," in *Proc. Roy. Soc. London. Ser. A: Math. Phys. Sci.*, vol. 439, no. 1907, pp. 553–558, 1992, doi: 10.1098/rspa.1992.0167.

[13] A. Montanaro, "Quantum walk speedup of backtracking algorithms," *Theory Comput.*, vol. 14, pp. 1–24, 2018, doi: 10.4086/toc.2018.v014a015.

[14] J. Zhao, "Possible implementations of oracles in quantum algorithms," *J. Phys.: Conf. Ser.*, vol. 2386, no. 1, 2022, Art. no. 012010, doi: 10.1088/1742-6596/2386/1/012010.

[15] J. M. Henderson, E. R. Henderson, A. Sinha, M. A. Thornton, and D. M. Miller, "Automated quantum oracle synthesis with a minimal number of qubits," *Proc. SPIE*, vol. 12517, 2023, Art. no. 1251706, doi: 10.1117/12.2663240.

[16] J. Sanchez-Rivero, D. Talavan, J. Garcia-Alonso, A. Ruiz-Cortes, and J. Murillo, "Automatic generation of an efficient less-than oracle for quantum amplitude amplification," in *Proc. IEEE/ACM 4th Int. Workshop Quantum Softw. Eng.*, 2023, pp. 26–33, doi: 10.1109/Q-SE59154.2023.00011.

[17] M. Grassl, B. Langenberg, M. Roetteler, and R. Steinwandt, "Applying Grover's algorithm to AES: Quantum resource estimates," in *Post-Quantum Cryptography*, T. Takagi, Ed. Cham, Switzerland: Springer, 2016, pp. 29–43, doi: 10.1007/978-3-319-29360-8_3.

[18] R. H. Preston, "Applying Grover's algorithm to hash functions: A software perspective," *IEEE Trans. Quantum Eng.*, vol. 3, 2022, Art. no. 2500710, doi: 10.1109/TQE.2022.3233526.

[19] D. Joseph, A. J. Martinez, C. Ling, and F. Mintert, "Quantum mean-value approximator for hard integer-value problems," *Phys. Rev. A*, vol. 105, no. 5, 2022, Art. no. 52419, doi: 10.1103/PhysRevA.105.052419.

[20] N. Bindel, X. Bonnetain, M. Tiepelt, and F. Virdia, "Quantum lattice enumeration in limited depth," *Cryptol. ePrint Arch.*, 2023, doi: 10.1007/978-3-031-68391-6_3.

[21] S. Bai, M.-I. van Hoof, F. B. Johnson, T. Lange, and T. Ngo, "Concrete analysis of quantum lattice enumeration," in *Proc. Int. Conf. Theory Appl. Cryptol. Inf. Secur.*, 2023, pp. 131–166, doi: 10.1007/978-981-99-8727-6_5.

[22] Y. Chen and P. Q. Nguyen, "BKZ 2.0: Better lattice security estimates," in *Proc. Int. Conf. Theory Appl. Cryptol. Inf. Secur.*, 2011, pp. 1–20, doi: 10.1007/978-3-642-25385-0_1.

[23] M. R. Albrecht, L. Ducas, G. Herold, E. Kirshanova, E. W. Postlethwaite, and M. Stevens, "The general sieve kernel and new records in lattice reduction," in *Proc. Annu. Int. Conf. Theory Appl. Cryptographic Techn.*, 2019, pp. 717–746, doi: 10.1007/978-3-030-17656-3_25.

[24] A. K. Lenstra, H. W. Lenstra, and L. Lovász, "Factoring polynomials with rational coefficients," *Math. Ann.*, vol. 261, pp. 515–534, 1982, doi: 10.1007/BF01457454.

[25] C.-P. Schnorr, "A hierarchy of polynomial time lattice basis reduction algorithms," *Theor. Comput. Sci.*, vol. 53, no. 2/3, pp. 201–224, 1987, doi: 10.1016/0304-3975(87)90064-8.

[26] G. Hanrot, X. Pujol, and D. Stehlé, "Analyzing blockwise lattice algorithms using dynamical systems," in *Proc. Int. Conf. Theory Appl. Cryptol. Inf. Secur.*, 2011, pp. 447–464, doi: 10.1007/978-3-642-22792-9_25.

[27] J. Li and P. Q. Nguyen, "A complete analysis of the BKZ lattice reduction algorithm," *Cryptol. ePrint Arch.*, 2020. [Online]. Available: https://ia.cr/2020/1237

[28] J. Kelner, "An algorithmist's toolkit, lecture 19," Massachusetts Institute of Technology (MIT), Nov. 2009. [Online]. Available: https://ocw.mit.edu/courses/mathematics/18-409-algorithmists-toolkit-fall-2009/lecture-notes/

[29] M. S. A. Mamun and D. Menville, "Quantum cost optimization for reversible sequential circuit," 2014, *arXiv:1407.7098*, doi: 10.48550/arXiv.1407.7098.

[30] H. Thapliyal and N. Ranganathan, "Design of efficient reversible logic-based binary and BCD adder circuits," *ACM J. Emerg. Technol. Comput. Syst.*, vol. 9, no. 3, pp. 1–31, 2013, doi: 10.1145/2491682.

[31] H.-S. Li, P. Fan, H. Xia, H. Peng, and G.-L. Long, "Efficient quantum arithmetic operation circuits for quantum image processing," *Sci. China Phys., Mech. Astron.*, vol. 63, pp. 1–13, 2020, doi: 10.1007/s11433-020-1582-8.

[32] H.-S. Li, P. Fan, H. Xia, and G.-L. Long, "The circuit design and optimization of quantum multiplier and divider," *Sci. China Phys. Mech. Astron.*, vol. 65, no. 6, Jun. 2022, Art. no. 260311, doi: 10.1007/s11433-021-1874-2.

[33] E. Muñoz-Coreas and H. Thapliyal, "T-count optimized design of quantum integer multiplication," 2017, *arXiv:1706.05113*, doi: 10.48550/arXiv.1706.05113.

[34] J. Nie, Q. Zhu, M. Li, and X. Sun, "Quantum circuit design for integer multiplication based on Schönhage–Strassen algorithm," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 42, no. 12, pp. 4791–4802, Dec. 2023, doi: 10.1109/TCAD.2023.3279300.

[35] H. Thapliyal, "Mapping of subtractor and adder-subtractor circuits on reversible quantum gates," in *Transactions on Computational Science XXVII*, M. Gavrilova and C. Tan, Eds. Lecture Notes in Computer Science, vol 9570. Berlin, Germany: 2016, pp. 10–34, doi: 10.1007/978-3-662-50412-3_2.

[36] Qiskit contributors, "Qiskit: An open-source framework for quantum computing," IBM Quantum Development Team, 2023. Accessed: Oct. 12, 2024. doi: 10.5281/zenodo.2573505

[37] A. S. Green, P. L. Lumsdaine, N. J. Ross, P. Selinger, and B. Valiron, "Quipper: A scalable quantum programming language," *ACM SIGPLAN Notices*, vol. 48, no. 6, pp. 333–342, 2013, doi: 10.1145/2499370.2462177.

[38] P. Fu, K. Kishida, N. J. Ross, and P. Selinger, "Proto-quipper with dynamic lifting," *Proc. ACM Program. Lang.*, vol. 7, pp. 309–334, 2023, doi: 10.1145/3571204.

[39] A. JavadiAbhari et al., "ScaffCC: A framework for compilation and analysis of quantum computing programs," in *Proc. 11th ACM Conf. Comput. Front.*, 2014, pp. 1–10, doi: 10.1145/2597917.2597939.

[40] S. Wang et al., "Quantum circuits design for evaluating transcendental functions based on a function-value binary expansion method," *Quantum Inf. Process.*, vol. 19, no. 10, pp. 1–31, 2020, doi: 10.1007/s11128-020-02855-7.

[41] S. Wang, Z. Wang, W. Li, L. Fan, Z. Wei, and Y. Gu, "Quantum fast Poisson solver: The algorithm and complete and modular circuit design," *Quantum Inf. Process.*, vol. 19, no. 6, pp. 1–25, 2020, doi: 10.1007/s11128-020-02669-7.

[42] T. Hoefler, T. Häner, and M. Troyer, "Disentangling hype from practicality: On realistically achieving quantum advantage," *Commun. ACM*, vol. 66, no. 5, pp. 82–87, 2023, doi: 10.1145/3571725.