Variational quantum solutions to the Shortest Vector Problem

Citation for published version:

Digital Object Identifier (DOI):
10.22331/q-2023-03-02-933

Link:
Link to publication record in Edinburgh Research Explorer

Document Version:
Publisher's PDF, also known as Version of record

Published In:
Quantum

General rights
Copyright for the publications made accessible via the Edinburgh Research Explorer is retained by the author(s) and / or other copyright owners and it is a condition of accessing these publications that users recognise and abide by the legal requirements associated with these rights.

Take down policy
The University of Edinburgh has made every reasonable effort to ensure that Edinburgh Research Explorer content complies with UK legislation. If you believe that the public display of this file breaches copyright please contact openaccess@ed.ac.uk providing details, and we will remove access to the work immediately and investigate your claim.
A fundamental computational problem is to find a shortest non-zero vector in Euclidean lattices, a problem known as the Shortest Vector Problem (SVP). This problem is believed to be hard even on quantum computers and thus plays a pivotal role in post-quantum cryptography. In this work we explore how (efficiently) Noisy Intermediate Scale Quantum (NISQ) devices may be used to solve SVP. Specifically, we map the problem to that of finding the ground state of a suitable Hamiltonian. In particular, (i) we establish new bounds for lattice enumeration, this allows us to obtain new bounds (resp. estimates) for the number of qubits required per dimension for any lattices (resp. random $q$-ary lattices) to solve SVP; (ii) we exclude the zero vector from the optimization space by proposing (a) a different classical optimisation loop or alternatively (b) a new mapping to the Hamiltonian. These improvements allow us to solve SVP in dimension up to 28 in a quantum emulation, significantly more than what was previously achieved, even for special cases. Finally, we extrapolate the size of NISQ devices that is required to be able to solve instances of lattices that are hard even for the best classical algorithms and find that with $\approx 10^3$ qubits such instances can be tackled.

1 Introduction

Cryptography studies the limits of computing: what can and cannot efficiently be computed. In 1976 Diffie and Hellman significantly expanded the realm of the possible by inventing public key cryptography [1] which allows two parties to agree on a shared secret over a public channel in the presence of a wiretapping adversary. Since its invention public-key cryptography has seen widespread adoption and is now a crucial building block for securing, say, the Internet. However, this advance was not unconditional but relies on the presumed hardness of some computational problem. Virtually all currently deployed public-key encryption schemes rely on the difficulty of one of two computational problems: the discrete logarithm problem and the problem of factoring large integers.

Everything changed in 1994 when Peter Shor’s seminal work [2] showed that quantum computers could effectively solve those two central problems. While the timeline for when sufficiently big quantum computers may be available is uncertain, the proposed such timelines and the threat of “collect-now-decrypt-later”-style attacks provoked global efforts to develop “post-quantum cryptography”, cryptographic schemes that run on classical computers but resist attacks with quantum computers. The centre of these international efforts is the Post-Quantum Standardisation Process (NIST PQC) by the US National Institute of Standards and Technology (NIST) [3]. To date, there are several candidates for post-quantum cryptography, mainly lattice-based, code-based, hash-based, multivariate cryptography and supersingular elliptic curve isogeny cryptography. Lattice-based cryptography seems to be a prime contender for large scale adaption: among the winners the NIST PQC process, three out of four are based on lattices.

While, of course, all post-quantum candidates are conjectured to be hard also on a quantum computer, a pressing question for adoption is “how hard?”. That is, in order to pick parameters that are both efficient in practice but quantifiably resist attacks, the research community studies the quantum resources required to solve the underlying hard problems.

On the other hand, it is expected that the transition to quantum computers will start with a phase referred to as Noisy Intermediate Scale Quantum (NISQ), featuring devices that consist of at most one thousand erroneous qubits. This low number of qubits makes use of any known error-correction technique infeasible, something that also puts stringent restrictions on the depth a quantum computation can have before the noise becomes dominant leading to essentially random outcomes. To overcome this limitation of these devices, hybrid classical-quantum algorithms are designed specifically for these devices while most of the famous quantum algorithms like Grover’s search,
Quantum Fourier Transform or Shor’s algorithms are impracticable. A natural question is thus “how hard are lattice problems on NISQ devices?”.

This question sheds light on the performance of these devices on a natural and central computational problem.

Contributions. After some preliminaries (Section 2), we determine a suitable mapping of the central hard problem on lattices, the Shortest Vector Problem (SVP) into the ground state of a Hamiltonian operator, the form required by the classical-quantum optimisation algorithms of our interest. In Section 3, we introduce Ising spin Hamiltonian operators and explain how to map SVP to an optimisation problem in this framework. We discuss the challenges that arise with this formulation and give a solution to the “zero vector problem” for VQE.

We also perform extensive classical numerical experiments or to impose a NISQ quantum algorithm to compute an HKZ reduced basis, one of the strongest notion of reduction, using \( \frac{1}{2}n \log_2 n + O(n) \) qubits and thus solving SVP in passing. This cost of non-adaptive enumeration was previously known only for a special class of lattices. We also perform extensive classical numerical experiments to study the average case behaviour of lattice reduction in the context of our NISQ quantum enumeration algorithm. In Section 5 we then show that our bounds allow us to run quantum emulation experiments that using up to 28 qubits are able to solve SVP in dimension 28 which is considerably more than prior literature. Extrapolating our experimental data we find that between 1,000 and 1,600 qubits suffice to encode SVP for a dimension 180 lattice, the current record dimension for “Darmstadt SVP Challenge” [4].

For the avoidance of doubt, our results do not violate any previous claims on the hardness of lattice problems on quantum computers because in general we may hope for a running time at best \( 2^{n/2 + o(n)} \) for instances encoded in \( \lambda = \frac{1}{2} n \log_2 n + O(n) \) qubits.

2 Preliminaries

Lattices. A (Euclidean) lattice \( \mathcal{L} \) is a discrete subgroup of \( \mathbb{R}^d \), or equivalently the set \( \mathcal{L}(b_1, \ldots, b_n) = \{ \sum_{i=1}^n x_i \cdot b_i : x_i \in \mathbb{Z} \} \) of all integer combinations of \( n \) linearly independent vectors \( b_1, \ldots, b_n \in \mathbb{R}^d \). Such \( b_i \) form a basis of \( \mathcal{L} \). We say that a matrix \( B \) forms a basis of \( \mathcal{L} \) if its rows form a basis of \( \mathcal{L} \). All the bases have the same number \( n \) of elements, called the dimension or rank of \( \mathcal{L} \). The dual \( \hat{\mathcal{L}} \) of a lattice \( \mathcal{L} \) is the set of all vectors \( x \in \text{span}(\mathcal{L}) \) such that \( \langle x, y \rangle \) is an integer for all \( y \in \mathcal{L} \). If \( B \) is a basis of \( \mathcal{L} \) then \( \hat{B} = (B \cdot B^T)^{-1} \cdot B \) is a basis of \( \hat{\mathcal{L}} \). We call \( \hat{B} \) the “dual basis” of \( B \).

Lattice algorithms often involve the orthogonal projections \( \pi_i : \mathbb{R}^n \rightarrow \text{span}(b_1, \ldots, b_{i-1})^\perp \) for \( i = 1, \ldots, n \). The Gram–Schmidt orthogonalisation (GSO) of \( B \) is \( B^* = (b_1^*, \ldots, b_n^*) \), where the Gram–Schmidt vector \( b_i^* \) is \( \pi_i(b_i) \).

One of the fundamental algorithmic problems related to lattices is to find a shortest non-zero element of an arbitrary lattice (with respect to its Euclidean norm), given an arbitrary basis of this lattice. This problem is referred to as the shortest vector problem (SVP) and the length of such a vector denoted by \( \lambda_1(\mathcal{L}) \). It is a central premise of lattice-based cryptography that solving SVP (and its decision variant GapSVP) within a polynomial factor takes super-polynomial time also on a quantum computer [5].

It is well-known that the hardness of the SVP is related to the “quality” of the input basis which informally quantifies the length of the vectors in the basis and the angles between. Intuitively, a basis with short and relatively orthogonal vectors is of higher quality. Therefore, a fundamental problem in lattice-based cryptography is to increase the quality of a given basis, a process known as lattice reduction. The celebrated LLL algorithm [6] was the first polynomial-time algorithm that computes a reduced basis of guaranteed quality, namely the first vector is at most exponentially longer than the shortest vector of the lattice. The BKZ algorithm [7] is a generalisation of LLL to obtain more strongly reduced basis at the expense of a higher running time. More precisely, the BKZ algorithm requires us to choose a so-called block size \( \beta \): the larger the \( \beta \), the stronger the reduction but the higher the running time (which is at least exponentially longer than the shortest vector of the lattice).

The BKZ algorithm is critical to understanding the complexity of BKZ and lattice reduction. This, in turns, is critical to choosing security parameters of cryptographic primitives [5].

Classically, SVP can be solved in time \( n^{1/(2d)+o(1)} \) and poly\( (n) \) memory or \( 2^{O(d \log n + \log \log n)} \) time and memory.
When $\beta = n$ we recover the HKZ algorithm producing HKZ reduced bases, which one of the strongest notion of reduction [9]. The first vector of such a basis is always a shortest vector of the lattice. Furthermore, HKZ basis naturally lend themselves to be computed recursively and enjoy many good properties (see Section 4.2 for more details), especially in conjunction with enumeration [10]. Enumeration algorithms list all of the lattice points within a ball of prescribed radius $r$. One of the most important aspect of enumeration is to correctly chose the enumeration radius $r$ so that it is larger than $L_1(L)$, but not too large since the running time increases rapidly with $r$. For random lattices, the so-called Gaussian Heuristic gives a good estimate of $L_1(L)$ as $gh(L) \approx \sqrt{\frac{2\pi}{\lambda_1}} \cdot \det(L)^{1/n}$.

The fastest known (heuristic) quantum algorithm [11] for solving SVP is a “sieving algorithm” runs in time $2^{0.257n} + o(n)$, uses QRAM [12] of maximum size $2^{0.075n} + o(n)$, a quantum memory of size $2^{0.045n} + o(n)$ and a classical memory of size $2^{0.2075}n + o(n)$. The second main class of quantum algorithms for solving SVP are “lattice-point enumeration” algorithms which combine exhaustive search with projections [13]. These algorithms run in time $n^{\frac{1}{4}}(4e)^{+o(n)} + poly(d)$ memory. In many cryptography applications $n^{\frac{1}{4}+o(n)}$ seems plausible [14].

Variational Quantum Algorithms. (VQAs) [15] are hybrid classical-quantum algorithms that can solve a variety of problems, including optimisation problems. VQAs are believed to be one of the most promising approaches to use and possibly offer quantum advantage on near-term quantum devices – also known as Noisy Intermediate-Scale Quantum (NISQ) devices. The quantum devices that currently exist and those that will become available in the coming 2-5 years are expected to have at most 1000 qubits (intermediate-scale) and have imperfections (noisy).

The optimiser then calculates new parameter $\theta$ and the procedure repeats until some stopping criterion is reached and an estimate for the ground state is produced. For our VQE runs we used hardware efficient ansätze. QAOA is by definition a problem specific ansatz since the family is constructed as a discretised version of Quantum Adiabatic Computation. This puts a constraint on the form of the Hamiltonian (which makes it harder to solve SVP with fewer qubits see Section 3.2), but has the theoretical guarantee that for sufficiently deep circuits the solution should be found. Specifically, for Hamiltonian measurement is run multiple times and the output is fed to a classical optimisation part that essentially off-loads part of the computation to a classical device. VQAs appear to offer advantages over classical algorithms in various areas of quantum chemistry [16, 17] and is a promising approach for many other areas including combinatorial optimization [18], quantum machine learning [19] and quantum metrology [20].

Since SVP can be formulated as an optimisation problem, we focus on two most widely used VQAs for combinatorial optimisation, namely Variational Quantum Eigensolver (VQE) [21, 22] and Quantum Approximate Optimisation Algorithm (QAOA) [23]. The first step for both of them, as well as for quantum annealing [24], is to encode the problem (SVP here) to the ground state (smallest eigenvalue) of a Hamiltonian operator $\mathcal{H}$. For QAOA and quantum annealing, the Hamiltonian needs to be an Ising spin Hamiltonian (i.e. involving only Z spins and up to quadratic interaction terms) and for quantum annealing extra limitations due to the connectivity of the spins apply. For VQE the Hamiltonian can take more general form (including higher order terms and/or X spins). The second step, once the Hamiltonian is chosen, is to select an ansatz state $\psi(\theta)$ – a family of quantum states, parameterised by $\theta$, that are the output of a simple parameterised circuit. Ideally, we would like to ensure that elements of the family considered are close to the ground state of the problem’s Hamiltonian. The two classes of ansätze that exist are the hardware efficient ones that are essentially chosen for the ease that can be implemented at a given quantum hardware, and the problem specific that are ansätze that use information about the problem for example using the problem’s Hamiltonian. The former are less prone to errors and can be used with any Hamiltonian but have no guarantee to be “dense” around the true ground state, while the latter are more sensitive to noise and can be used with specific classes of Hamiltonians but are designed to have states close to the true ground state. The third step is to prepare a state from the ansatz and measure it where this step is repeated multiple times. From these repetitions an estimate of the expectation $\langle \psi(\theta) | \mathcal{H} | \psi(\theta) \rangle$ is calculated and passed as the cost value for the choice of parameters to a classical optimiser.
ans that involve only spin $Z$ terms, to compute the energy/cost $C(\theta)$ of a quantum state $|\psi(\theta)\rangle$ we prepare the state and measure in the computational basis $N$ times. Each run gives outcomes (bit-strings) $x_i$, and for every outcome we compute the related cost $m_i(x_i)$. Our estimate of the cost of the state is $C(\theta) = |\langle \psi(\theta)|H|\psi(\theta)\rangle| \approx \frac{1}{N} \sum_{i=1}^{N} m_i$ and is used by the classical optimiser. For classical combinatorial optimization problems one can find that other methods are performing better. The Conditional Variate at Risk (CVaR) [25] and the Ascending-CVaR [26] are two methods that give better results and in this work we will use the former. In [25] instead of computing the cost taking the average of the values $m_i$, they considered ordering the values from the smallest to the larger and counting the $\alpha$-tail of the distribution. Specifically, $\alpha$ is to be chosen heuristically and the cost is calculated as an average of $[\alpha N]$ lowest measurements outcomes. Suppose $\{\tilde{m}_i\}_{i=1,...,N}$ is a sorted set of $\{m_i\}_{i=1,...,N}$ in non-decreasing order. Then the cost is calculated as: $CCVaR_{\alpha}(\theta) = \frac{1}{|C_N|} \sum_{i=1}^{[\alpha N]} \tilde{m}_i$.

We note that finding a ground state of a Hamiltonian is QMA-complete in general, but for specific Hamiltonians the ground can be found efficiently – adiabatic quantum computing is a universal model, that finds efficiently the ground state of those Hamiltonians that correspond to BQP problems. Indeed, in our case, since SVP is believed to be outside BQP we do not expect to find the solution efficiently. On the other hand, obtaining a polynomial speed-up is valuable for the cryptanalysis – after all Grover’s algorithm also provides such a moderate speed-up. In particular, we do not expect to do better than Grover – query complexity bounds indicate that for our problem we can get at most a small constant improvement to Grover. However, due to the fact that our approach is heuristic, we may still be able to get considerable (larger) speed-up for certain (but not the hardest) instances. To properly analyse the time-complexity of a variational quantum algorithm, one needs to bound the scaling of the probability that the algorithm (at the end of the classical-quantum iterations) returns the correct solution – viewed differently, to bound the overlap of the output quantum state with the true shortest-vector. Then by repeating the algorithm sufficient number of times, one is guaranteed to find the correct solution with high probability. While such bounds have been found for certain problems [27, 23, 28], in the general case it is hard to obtain them (or even impossible), so this is left for future work. Note that the classical time-complexity scales exponentially, therefore even if this probability vanishes exponentially fast (as expected) our solution may still give competitive results offering similar performance to other “fault-tolerant” quantum algorithms.

2.1 Related Work

There have been several works focusing on translating SVP into Hamiltonian $H$ where the ground state corresponds to the shortest lattice vector. Note that we can trivially achieve encodings where the corresponding eigenvalues of $H$ define the order of lattice vector lengths. The resulting variational quantum formulation is thus capable of solving the approximate SVP, a relaxation of SVP, on which the security of most lattice-based protocols is relied upon. In [29] the energy gaps between the first three excited states of the problem Hamiltonian are analysed when solving low dimensional SVP via adiabatic computation. The ground state is by their construction a zero state and hence the first excited state is sought. The results suggest the existence of “Goldilocks” zones. In such zones the adiabatic evolutions are slow enough that there is a high probability of sampling any of the first three excited states without any strong dominance of any of them. As a consequence, in this case it is possible to obtain the shortest non-zero vector with a small number of measurement samples. This motivates the use of QAOA to find the ground state as it mimics the adiabatic computation. However, the experiments were performed only for lattices up to 4 dimensions. Moreover, as SVP is an integer optimisation problem, bounds on the ranges of the (integer) variables need to be defined prior to mapping the problem into a binary optimisation problem. The authors make an experimental guess that each of them grows linearly with lattice dimension resulting in guess of $O(n \log n)$ qubit requirement. The qubit requirement was later proved in [30] for special lattices with what they called “an optimal Hermite Normal Form (HNF)”.

Indeed, the matrix of a $q$-ary lattice is $[I_{n-k} \ X \ 0 ]$, where $X \in \mathbb{Z}_{q}^{(n-k) \times k}$ has rank $k$, is already in HNF.

\footnote{Here sufficient scales inversely with the probability of success of a single run}
3 Direct approaches to SVP with Variational Quantum Algorithms

Recall that the starting point for VQE and QA赵 (but also for quantum annealing) is to encode the problem to the ground state of a Hamiltonian $\mathcal{H}$. As in other combinatorial optimisation problems, one first needs to turn the problem to a quadratic unconstrained binary (QUBO) form and then there is a standard method to obtain an Ising Hamiltonian (i.e., a Hamiltonian that involves only $Z$ operators and the interaction terms involving just pairs of qubits). Note that to run VQE one can have more general Hamiltonians and as we will see later, in some cases it is simple to include constraints (or even higher order terms). Coming back to QUBO formulations the general form of the cost is $C(s_1s_2\ldots s_n) = c + \sum_i c_i s_i + \sum_{i\neq j} c_{ij} s_i s_j$, for string binary variables $s_1s_2\ldots s_n$, for some coefficients $c, (c_{ij})_{1\leq i,j \leq n}$. Since Pauli-$Z$ operators have $\pm 1$ eigenvalues one maps each binary variable $s_i$ to $\frac{1}{\sqrt{2}}(\mathbb{I} + z_i)$ leading to the Ising Hamiltonian, $H = \sum_i c_i \frac{1}{\sqrt{2}}(\mathbb{I} + z_i) + \sum_{i\neq j} c_{ij} \frac{1}{\sqrt{2}}(\mathbb{I} + z_i) \cdot \frac{1}{\sqrt{2}}(\mathbb{I} + z_j)$ where $z_i$ and $I_i$ are the Pauli-$Z$ and identity operators respectively, acting on $i$-th qubit.

3.1 Mapping the Shortest Vector Problem into QUBO formulation

A first approach (also starting point of [29, 30]), is to consider as variables the coefficients of the basis vectors. Each choice of the coefficients determines a vector in the lattice, and its length can be easily calculated using the basis matrix. There are two major challenges with this approach. The first one is that the coefficients are integers while we want to use a finite number of binary variables. Recall that each binary variable will result to a qubit so we can only use a finite number while making this number as small as possible is crucial to run it in a NISQ device. We therefore need to truncate (limit) the possible coefficients, but in a way that the shortest vector is included in our possible solutions. The second challenge is that if the variables are the coefficients then a possible vector is the all zero vector, a vector that clearly is shorter than the shortest (non-zero) vector we are searching. We would therefore need to impose the constraint $x \neq 0$ at some level.

Going back to the formulation, given an $n$-dimensional full-rank row-major lattice basis matrix $B$, we define lattice $\mathcal{L}(B) = \{x \cdot B|x \text{ row vector}, x \in \mathbb{Z}^n\}$. The shortest vector problem (SVP) finds the solution $\lambda_1 = \min_{y \in \mathcal{L}(B) \setminus \{0\}} \|y\|$. Let $x$ be a row vector of coefficients and $G = B \cdot B^T$ a Gram matrix of the basis. Then $y = x \cdot B \implies \|y\|^2 = x \cdot B \cdot B^T \cdot x^T = x \cdot G \cdot x^T$ which allows us to reformulate it as a quadratic constrained integer optimisation problem:

$$\lambda_1^2 = \min_{y \in \mathcal{L}(B) \setminus \{0\}} \|y\|^2 \quad (1)$$

As a consequence of rank-nullity theorem, we have $\lambda_1 \neq 0$ as required. In order to convert (1) into a binary optimisation problem, we need bounds $|x_i| \leq a_i$ for all $i = 1, \ldots, n$. This has been the core problem of all approaches to map the SVP into a Hamiltonian and we provide tighter and more general bounds in Section 4.

We now define new binary variables $\{\tilde{x}_{ij}\}_{0 \leq j \leq \lfloor \log 2a \rfloor}$ using the integer variables $x_i$ and the bound $a_i$:

$$|x_i| \leq a \implies \sum_{j=0}^{\lfloor \log 2a \rfloor-1} 2^j \tilde{x}_{ij} + (2a + 1 - 2^{\lfloor \log 2a \rfloor}) \cdot \tilde{x}_{i, \lfloor \log 2a \rfloor} \sum_{j=0}^{\lfloor \log 2a \rfloor-1} 2^j \tilde{x}_{ij} + (2a + 1 - 2^{\lfloor \log 2a \rfloor}) \cdot \tilde{x}_{i, \lfloor \log 2a \rfloor} \implies$$

where the last term ensures we do not enlarge the search space. By substituting into (1) and ignoring the $x \neq 0$ constraint, the resulting QUBO becomes (3) where $c$, $c_{ij}$ and $d_{ij,k,l}$ are calculated constants resulting from the substitution.

$$\min_{\tilde{x}_{1,0}, \ldots, \tilde{x}_{1, \lfloor \log 2a \rfloor}, \ldots, \tilde{x}_{n,0}, \ldots, \tilde{x}_{n, \lfloor \log 2a \rfloor}} \sum_{i,j} c_{ij} \tilde{x}_{ij} + \sum_{i,j,k,l} d_{ij,k,l} \tilde{x}_{ij} \tilde{x}_{k,l}$$

3.2 Encoding the $x \neq 0$ constraint

In deriving Equation (3) we ignored the $x \neq 0$ condition in the definition of SVP. The true minimum is zero and is obtained by setting all the integer variables $x_1, \ldots, x_n$ to zero. Instead we want to obtain the second smallest value. In other words, we are seeking the first excited state of the corresponding Hamiltonian, with the extra information that the “true” ground state (zero vector) is known. There are three ways to address this. First is to ignore the constraint, run the optimisation to find the ansatz state with greatest overlap with the zero vector, and hope that it has (ideally large) overlap with the first excited state [29]. Even if this approach succeeds in finding SVP for small dimensions it is unlikely to work well at large scales. The second solution is to exclude the zero vector by imposing it in a form of a constraint that “penalises” the zero vector. To map the problem back to a QUBO, however, requires introducing extra variables ($n - 2$ in our case), making this approach less practical. Since this approach can work both for QAOA and quantum annealing where the last and more practical approach does not work, we give the details of this mapping in Appendix A.
The third approach, that we will analyse here and used in our numerical experiments, can be used in VQE. One directly targets the first excited state by modifying the classical loop of the VQE i.e. modifying the way that the cost is evaluated. In the general form, assuming that |ψ⟩ is the ground state, instead of using the expression \( C(\psi) = \langle \psi | H | \psi \rangle \) for the cost, we use \( C(\psi) = \langle \psi | H | \psi \rangle \frac{1}{1-|\langle \psi | \omega \rangle|^2} \), where the multiplicative factor is the inverse of the probability of not being in the ground state. This gives infinite cost to the ground state, and in general penalises states not being in the ground state. This gives infinite cost to the ground state, and in general penalises states with greater overlap with the ground state. It is easy to see that this cost is minimised at the first excited state of the original Hamiltonian. Note that other approaches to find the first excited state exist (see for example [32]).

In our case the ground state is the zero vector. Let \( N \) be the cardinality of \( m_i \)'s (measured bit-strings) that are non-zero. Then \( \frac{N}{N} \) is the numerical estimate of the inverse of the probability not being the zero vector. Therefore we will be using the modified cost \( C'(\theta) = \left( \frac{x}{N} \right)^2 \sum_{i=1}^{N} m_i \). Equivalently, this means that to compute the cost of a state, we disregard the zero-vector outcomes taking the average value over all the other outcomes. In the measurements of the final ansatz state, at the end of the optimisation, we output the measurement sample with the lowest non-zero energy. The advantage of this approach is that the quantum part (states and measurements) are identical to the unconstrained one, and the only difference is in the way that the measurement outcomes are used to assign a cost to different states in the classical optimisation part of the VQE.

### 3.3 Handling approximate solutions

As the dimension of the lattice increases, the probability that the algorithm converges to a shortest vector might become vanishingly small. Instead, the system might return a short-but-not-shortest vector. In this case, one approach would be to replace a vector of the basis by the newly obtained one, making the basis more reduced, and restart. This approach has been experimentally observed to work in [33]. In the classical setting, the use of tightly controlled approximate SVP oracles was shown to provide an exponential speed-up over enumeration with exact SVP oracles [34], but here we likely do not get to choose the looseness of our oracles. In principle, if we could characterise the distribution of the output of the algorithm and quantify the expected size of the vector, we could estimate the number of times the algorithm needs to be re-run before obtaining a shortest vector with good probability. We leave this analysis as future work since it is nontrivial to estimate the distribution of the output of the algorithm.

### 4 Bounds on lattice enumeration and application to variational quantum algorithms for solving SVP

In the previous section we gave a map of SVP to a QUBO formulation that is suitable for VQE. This map, however, relies on bounds on each \(|x_i|\). In this section, we obtain worst case bounds on the \(|x_i|\) based on the orthogonality defect of the dual basis. We then give two applications of this result. First, we show how to obtain a recursive algorithm to compute a HKZ reduced basis and thus solving the SVP in passing. This method gives us the best asymptotical bound on the number of qubits required. Second, we estimate the number of qubits required to directly solve the SVP by reducing the dual basis with recursive or classical preprocessing and then applying the NISQ enumeration.

#### 4.1 General bounds on lattice enumeration

As we have seen in Section 3.1, we need to choose a bound on each of the \(|x_i|\) to obtain a finite optimisation problem. This bound will then determine the number of qubits required. In order to be sure to find a solution, we first derive a general bound on all the \(|x_i|\) that correspond to lattice points in a ball of radius \(A\). We then use the Gaussian heuristic to choose the radius \(A\) so as to ensure that this ball contains at least one nonzero vector for most lattices and therefore a shortest vector.

**Lemma 1.** Let \( x_1, \ldots, x_n \) be such that \(|x_1 \cdot b_1 + \cdots + x_n \cdot b_n| \leq A\), then for all \( i = 1, \ldots, n \) we have \(|x_i| \leq A \cdot \|\hat{b}_i\|\) where \(\hat{b}_1, \ldots, \hat{b}_n\) are the rows of \(\hat{B}\), a dual basis of \(B\) which is the matrix whose rows are \(b_1, \ldots, b_n\).

**Proof.** Let \(\delta_{i,j}\) denote the Kronecker delta. Observe that \(\langle b_i, \hat{b}_j \rangle = \delta_{i,j}\) for all \(i,j\). Indeed, \(\langle b_j, \hat{b}_i \rangle = (B \cdot \hat{B}^T)_{i,j} = (I_n)_{i,j} = \delta_{i,j}\). Now let \(v = x_1 \cdot b_1 + \cdots + x_n \cdot b_n\) for some \(x_1, \ldots, x_n \in \mathbb{Z}\) be such that \(\|v\| \leq A\). Then for any \(i\), we have that \(\langle v, \hat{b}_i \rangle = |x_i|\). But on the other hand, \(\langle v, \hat{b}_i \rangle \leq \|v\| \cdot \|\hat{b}_i\|\) which proves the result.

If we now make use of the Gaussian heuristic to choose the radius \(A\) and take \(A := \text{gh}(\mathcal{L}) = \sqrt{n/2\pi e} \cdot \text{vol}(\mathcal{L})^{1/n}\). The total number of qubits that we need will be

\[
N = \sum_{i=1}^{n} \left( \lfloor \log_2(2m_i) \rfloor + 1 \right) \leq 2n + \log_2 \prod_{i=1}^{n} m_i \quad (4)
\]

where \(m_i\) is the bound on the \(|x_i|\). Using the bound
of Lemma 1 we obtain
\[
\prod_{i=1}^{n} m_i \leq \left( \frac{n}{2\pi e} \right)^n \cdot \frac{\text{vol}(L)}{\prod_{i=1}^{n} \|b_i\|} = \left( \frac{n}{2\pi e} \right)^n \cdot \frac{\text{vol}(\hat{L})}{\prod_{i=1}^{n} \|b_i\|} = \left( \frac{n}{2\pi e} \right)^n \cdot \delta(\hat{L})
\]
where \( \hat{L} \) is the dual of \( L = L(b_1, \ldots, b_n) \) and \( \delta(\cdot) \) denotes the orthogonality defect. This suggests that the critical factor for the enumeration is the reduction of the dual of the lattice and not the lattice itself.

**Corollary 1.** The number of qubits required for the enumeration on the basis \( B \), assuming the Gaussian heuristic is bounded by
\[
2n + \log_2 \left( \left( \frac{n}{2\pi e} \right)^{n/2} \cdot \delta(\hat{B}) \right)
\]
where \( \delta(\cdot) \) denotes the orthogonality defect and \( C \) the multiplicative factor used with the Gaussian heuristic\(^7\).

### 4.2 Bound on the number of qubits for producing a HKZ basis

In this section, we show that our NISQ enumeration procedure can be used to HKZ reduce a basis in a recursive way using only \( \tilde{O}(n) \log(n) + O(n) \) qubits, thus solving SVP in passing. This is asymptotically the same as the algorithm in [30]. However, our algorithm works for any lattice, whereas the algorithm in [30] only works for lattices that have what they called “an optimal Hermite Normal Form”. See Section 2.1 for more details.

Recall that the notion of HKZ-reduced basis can be defined inductively as follows. Any vector \( b \) is a HKZ-reduced basis of \( L(b) \). A basis \( B = (b_1, \ldots, b_k) \), for \( k \geq 2 \), is HKZ-reduced if

- \( b_1 \) is a shortest vector in \( L(b) \),
- \( (\pi_2(b_2), \ldots, \pi_2(b_k)) \) is HKZ-reduced, where \( \pi_2(\cdot) \) is the projection on \( b_1^\perp \),
- \( \|b_2, b_1\| \leq \frac{1}{4}\|b_1, b_1\| \).

Intuitively, the first two conditions mean that \( b_1 \) is a shortest vector of the lattice \( L(b_1, \ldots, b_k) \), and after projecting on \( b_1^\perp \), \( \pi_2(b_2) \) is a shortest vector of the projected lattice \( L(\pi_2(b_2), \ldots, \pi_2(b_k)) \), and so on. This notion of reduction is much stronger than LLL-reduced basis and HKZ-reduced basis. The third condition is equivalent to the notion of size-reduction which is also required by LLL-reduced basis and is a technical condition.

We say that a basis \( B = (b_1, \ldots, b_n) \) is pseudo-HKZ-reduced if it is an LLL basis on which we apply an HKZ reduction to the first \( n - 1 \) vectors. This intuitively means that the basis is almost HKZ-reduced, except for the last vector, and the last vector is still controlled by the fact it comes from an LLL-reduction. In particular, the last vector is guaranteed to be at most exponentially longer than a shortest vector of the lattice.

Our algorithm is a variant of the classical algorithm for producing a HKZ reduced basis due to Kannan [9]. The main difference is that Kannan’s algorithm exclusively works on the (primal) basis whereas our algorithm has to reduce both the primal and the dual basis to control the number of qubits required in the enumeration steps. This difference requires us to work slightly differently. In particular, Kannan’s algorithm works by recursively producing a “quasi-HKZ” reduced basis, and then applying enumeration on this basis to obtain a full HKZ basis. A quasi-HKZ reduced basis is essentially a LLL basis on which we apply an HKZ reduction to the projection of the last \( n - 1 \) vectors (orthogonally to the first). In contrast, our algorithm first produces a pseudo-HKZ basis, i.e. an LLL-reduced basis on which we apply an HKZ reduction to the first \( n - 1 \) vectors. Our algorithm then runs enumeration on the dual.

**Algorithm 1:** HKZ-reduction algorithm using quantum enumeration

\[
\text{input} : \text{A basis} (b_1, \ldots, b_n) \\
\text{output} : \text{An HKZ basis of the same lattice}
\]

1. \( L := L(b_1, \ldots, b_n) \)
2. \( (d_1, \ldots, d_n) := \text{dual basis of} (b_1, \ldots, b_n) \)
3. \( (d_1, \ldots, d_n) := \text{LLL-reduce} (d_1, \ldots, d_n) \)
4. \( (d_1, \ldots, d_{n-1}) := \text{HKZ-reduce} (d_1, \ldots, d_{n-1}) \)
5. \( (b_1, \ldots, b_n) := \text{dual basis of} (d_1, \ldots, d_n) \)
6. Call the NISQ enumeration procedure proposed in Section 3 on \( (b_1, \ldots, b_n) \) using the bounds established in Lemma 1 and keep a shortest vector \( v \)
7. \( (b_1, \ldots, b_n) := \text{LLL-reduce} (v, b_1, \ldots, b_n) \)
8. Compute the orthogonal projections \( (b_2', \ldots, b_n') \) of \( (b_2, \ldots, b_n) \) on \( b_1^\perp \)
9. \( (b_2', \ldots, b_n') := \text{HKZ-reduce} (b_2', \ldots, b_n') \)
10. For each \( i \), set \( b_i := b_i' + \alpha_i b_1 \) where \( \alpha_i \in (-\frac{1}{2}, \frac{1}{2}) \) is such that \( b_i \) belongs to \( L \)
11. \text{return} \( (b_1, b_2, \ldots, b_n) \)

We will require the following well-known result about HKZ basis.

**Proposition 1 ([35]).** The orthogonality defect of a HKZ basis is bounded by \( \gamma_n^{n/2} \prod_{i=1}^{n} \sqrt{\pi^i} \) where \( \gamma_n \) is Hermite’s constant in dimension \( n \), and \( \gamma_n < \frac{1}{8} n + \frac{9}{8} \).

**Corollary 2.** The orthogonality defect of a HKZ basis
is \(\leq 2^{n \log_2 n - \left(\frac{n}{2} + \frac{1}{2} \log_2 e\right)n + O(\log n)}\).

Proof. See Appendix D. \(\square\)

**Theorem 1.** Assuming the Gaussian heuristic, Algorithm 1 produces an HKZ basis with \(\frac{3}{2} \pi \log_2(n) - 2.26n + O(\log(n))\) qubits.

Proof. We first show that the algorithm is correct by induction on \(n\). Let \(L = L(b_1, \ldots, b_n)\) at the beginning of the algorithm. First note that at the beginning of line 6, the basis \((b_1, \ldots, b_n)\) is still a basis of the original lattice \(L\) since all the previous operations preserve that property. Assuming the Gaussian heuristic is true (or true with a multiplicative factor\(^8\) \(C\)), the enumeration algorithm will indeed find a shortest nonzero vector \(v\) at line 6. The reduction at line 7 will allow to extract a basis of \(L\) among the \(n + 1\) vectors \(v, b_1, \ldots, b_n\). Since \(v\) is a shortest vector, the LLL reduction will ensure that \(b_1\) is a shortest vector of \(L\) at the beginning of line 8. Let \(L' = L(b_1', \ldots, b_n')\) be the projected lattice obtained after line 8. The reduction at line 9 does not change the lattice spanned by the \(b_i'\) which is now a HKZ basis. The lifting operation at line 10 ensures that \((b_1, \ldots, b_n)\) is a basis of \(L\) at the end of the algorithm. Since \(b_1\) is a shortest vector of \(L\) and \((b_2, \ldots, b_n)\) is HKZ and a basis of the projected lattice, it follows that the returned basis is HKZ.

We now analyse the qubits requirement of the algorithm. Let \(\bar{d}_1, \ldots, \bar{d}_n\) be the basis obtained after line 3 and \(\bar{d}_1', \ldots, \bar{d}_n'\) be its Gram-Schmidt orthogonalisation. By the properties of the LLL-reduction, we have \(\|\bar{d}_n\|^2 \leq 2^{n-1} \cdot \|\bar{d}_n'\|^2\) [37]. Let \((d_1, \ldots, d_n)\) be the basis obtained after line 4 and \(d_1', \ldots, d_n'\) be its Gram-Schmidt orthogonalization. Clearly \(d_n = \bar{d}_n\) since this vector was not touched by the HKZ reduction. It follows that \(d_n' = \bar{d}_n'\) because \(d_n'\) (resp. \(d_n'\)) is the projection of \(d_n = \bar{d}_n\) on the orthogonal of \(\text{span}(d_1, \ldots, d_n) = \text{span}(\bar{d}_1, \ldots, \bar{d}_n)\) which are equal because the HKZ reduction does not change the span. As a result, we have \(\|d_n\|^2 \leq 2^{n-1} \cdot \|d_n'\|^2\) after line 4 and therefore

\[
\delta(d_1, \ldots, d_n) = \delta(d_1, \ldots, d_{n-1}) \cdot \frac{\|d_n\|}{\|d_n'\|} = \frac{\|d_n\|}{\|d_n'\|} \leq 2^{n \log_2 n - \left(\frac{n}{2} + \frac{1}{2} \log_2 e\right)n + O(\log n)}
\]

by Corollary 2. Taking the dual of this basis at line 5 means at line 6, we are in a position to run the enumeration algorithm on a basis \((b_1, \ldots, b_n)\) whose dual \((d_1, \ldots, d_n)\) has a small orthogonality defect. By Corollary 1, the number of qubits required for this enumeration step is bounded by

\[
2n + \log_2 \left(\frac{C^2 n}{2\pi e}\right)^{n/2} \delta(d_1, \ldots, d_n)
\]

\[
\leq 2^{n \log_2 n} - \left(\frac{1}{2} \log_2 \pi + \log_2 e - \log_2 C\right)n + O(\log n)
\]

Denote by \(Q(n)\) the number of qubits necessary to run the algorithm in dimension \(n\). Since we run the enumerations sequentially, \(Q(n) = \max(Q(n-1), n \log_2(n) + O(n))\) and therefore \(Q(n) = 2^{n \log_2(n)} - 2.26n + O(\log(n))\) for \(C = 1\). \(\square\)

### 4.3 Solving the SVP directly in the NISQ era

As we have seen in the previous sections, the bound on the \(x_i\) and thus the running time of the enumeration fundamentally depend on the quality of the dual basis. The algorithm of the previous section takes advantage of this fact by recursively reducing the primal and dual basis and always running the enumeration on a dual quasi-HKZ reduced basis, one of the strongest notions of basis reduction. Unfortunately, the bound on the number of qubits that we obtained relies on worst case bounds on the orthogonality defect of such basis. In particular, the linear term in the bound of Theorem 1 is quite pessimistic for most bases.

In this section, we perform numerical experiments to understand the number of qubits necessary to run the quantum enumeration depending on the quality of the dual basis. As mentioned above, the cost of enumeration is affected by the quality of the input basis. We thus consider input bases preprocessed with LLL or BKZ-\(\beta\). Using Corollary 1, we draw Figure 1a which shows the number of qubits needed when the dual basis is LLL reduced or BKZ-\(\beta\) reduced with different \(\beta\). We choose the maximum value of \(\beta = 70\), since the running time becomes prohibitively long for larger \(\beta\). We also performed experiments with dual pseudo-HKZ reduced basis to understand the practical behavior of the algorithm in the previous section\(^9\). The graph was obtained by generating random \(q\)-ary lattices (for \(q = 65537\) and \(k = n/2\)), reducing them and then computing \(N\) as in (4) using the bounds in Lemma 1. Each experiment was repeated 5 times\(^10\) and the average was taken. We also performed regression using quadratic polynomials for LLL and BKZ reduced basis. This type of regression is expected to fit well since the number of qubits can be shown to be quadratic. On the other

\(^8\)Many papers and the SVP challenge [36], typically use the GH with a multiplicative factor \(C \approx 1.05\). For large \(n\), Minkowski’s bound on the shortest vector implies that \(C = \sqrt{2\pi e}\) actually holds but this is a much worse bound.

\(^9\)The \(q\)-ary lattices that we generate have fixed determinant, hence the number of qubits only depends on the orthogonality defect. We observed that the orthogonality defect of LLL and BKZ reduced basis varies little between runs, hence the small number of runs.
hand, as shown in Theorem 1, for pseudo-HKZ reduced dual basis, the number of qubits should grow as $\Theta(n \log_2(n))$ asymptotically. Hence, we fitted with a curve of form $\frac{3}{2}n \log_2(n) + an + b$.

In order to solve the current lattice approximate SVP record (dimension 180), we need approximately 1157 qubits when the dual is sufficiently reduced ($\beta = 70$ for example). This is considered to be achievable in the not so far future. We stress, however, that this does not imply such an algorithm will converge on a good solution efficiently but only that it can be run.

5 Experimental results

We run quantum emulation of the VQE algorithm on a classical computer to assess the performance of our SVP solver. Due to the limited number of emulable qubits we considered lattices with rank up to 28, much smaller than the cryptographically relevant ranks but much bigger than prior quantum attempts. While we estimated the space-complexity required to solve large dimensions, we cannot reliably extrapolate the performance (accuracy/time) due to the heuristic nature of VQE. The effects of noise and error-mitigation are beyond the scope of this work and are left for future explorations.

5.1 Experimental framework

For our experiments we first sample $A := \begin{pmatrix} I_{d-k} & \tilde{A} \\ 0 & q \cdot I_k \end{pmatrix} \in \mathbb{Z}^{d \times d}$ where $\tilde{A} \leftarrow \mathbb{Z}_{q^{(d-k)\times k}}$ and $I_x \in \mathbb{Z}^{x \times x}$ is the identity. We then consider the lattice $\mathcal{L}(A)$ spanned by the rows of this matrix. In particular, we first LLL reduce the entire lattice basis and then consider the sublattice of rank $n$ spanned by the first rows of the reduced basis. Our choice of $q$-ary lattices $\mathcal{L}(A)$ is partly motivated by their ubiquity in cryptography and by their connection to worst-case lattice problems. For example, finding short vectors in the dual of a random $q$-ary lattice is as hard as finding short vectors in any lattice [38]. We preprocess these lattices using the polynomial-time LLL algorithm for numerical stability reasons.

We extract a $n$-rank sublattice in dimension $d$ rather than considering full-rank lattices with $n = d$ since LLL will succeed in solving SVP for such small dimensions directly. The VQE receives a $n \times d$ matrix as input and this “leaves some work” for the quantum algorithm to do. Note that even though many instances of these sub-lattices were “almost” solved by the described procedure, it is little advantage for the VQE as the size of the search space was not reduced. Therefore, by using a random guess for initial optimization parameters of an ansatz circuit, our experiments represent scaled-down
real instances. Consequently, the hardness of SVP for VQE is in our experiments not even lowered if the shortest vector is already present in the given basis. The search space is defined by bounds on the coefficient vector as discussed in Section 4.1. However, the bounds there are worst-case and asymptotic. In contrast, lattice algorithms tend to perform much better on average than in the worst case, especially in small dimensions. Indeed, Figure 1a illustrates the same phenomenon here. We hence present a different qubit mapping strategy for which we compute probabilities of the shortest lattice vector lying within these bounds and in our quantum experiments we evaluate the probability of finding the ground state of problem Hamiltonian. The advantage is that the two problems: choosing an appropriate qubit mapping and finding the ground state of the problem Hamiltonian, can be tackled separately. The overall success probability of the variational SVP solver is then a product of the probability of encoding the shortest non-zero lattice vector in the search space and the probability of finding the Hamiltonian’s ground state.

Naive qubit mapping approach. Given an n-rank sublattice we assign one qubit per coefficient of the coefficient vector, i.e. \( |x_i| \leq 1 \) for \( 1 \leq i \leq n \). Using this approach we found that the shortest non-zero lattice vector was included in the search space with the following probabilities: rank 15: 80\%, rank 16: 75\%, rank 17: 75\%, rank 18: 74\%, rank 19: 71\%, rank 20: 70\%, rank 21: 64\%, rank 22: 62\%, rank 23: 57\%, rank 24: 55\%, rank 25: 50\%. See Appendix B for analysis of naive qubit mappings with ranks up to 50 that also considers different naive mapping strategies. The results presented in the appendix can be of interest for all experimental approaches to the SVP that use a Hamiltonian formulation.

Quantum emulation setup. We developed distributed variational quantum algorithms emulation software framework\(^{11}\) by modification of parts of Xacc\(^{11}\) library to utilize implementations of variational quantum algorithms and useful classical optimisers and we use QuEST\(^{40}\) for the underlying quantum emulation. The experiments have been run on Ngio 5, a distributed computer provided by Edinburgh Parallel Computing Centre featuring 2x Xeon Platinum 8260 24C 2.4GHz with 12x 16GB DRAM modules. The emulations use state-vector representation of intermediate quantum states and do not consider any effects of noise. Noise would affect the probability of finding the ground state (uncertainty of cost makes the classical optimization harder) (Figure 1b Top), but it would not affect the time requirements (Figure 1b Bottom), neither the number of qubits. As it has been noted, extrapolating our results to lattice dimensions of cryptographic interest is infeasible from our experiments, because the way the probability of success decays cannot be extrapolated (among other reasons because we cannot anticipate the effects that barren plateaus would have). This is why for our current contribution adding noise would not add much in our analysis, beyond perhaps, bringing lower the probabilities of solving the problem in the very low qubit experiments that are possible to emulate.

5.2 Experimental results

To improve the performance in our experiments we used CVaRa cost for VQE. We chose \( \alpha = 0.175 \) since this gave better results (see also Appendix C). We were able to solve the SVP with our emulator for lattices with rank up to 28. Our experiments with 128 instances suggest that the success probability of finding the shortest non-zero lattice vector remains roughly constant for lattice instances with ranks not much larger than 28. Figure 1b (top) depicts the averaged overlap of the final ansatz state with the ground state corresponding to the shortest non-zero lattice vector found by a classical enumeration. The overlap represents the probability of sampling the shortest lattice vector with a single measurement of the final ansatz state. From the figure we see that we need \( \approx 25 \) samples to obtain the solution. Moreover, we observed linear time scaling (see Figure 1b bottom). We note, however, that the ranks of cryptographically relevant lattices are larger (\( \approx 400 \)) and we cannot extrapolate our observations with confidence.

Acknowledgements

The authors acknowledge support from EPCC, including use of the NEXTGenIO system, which was funded by the European Union’s Horizon 2020 Research and Innovation programme under Grant Agreement no. 671951. The research of MA was supported by EPSRC grants EP/S020830/1, EP/S02087X/1 and by the European Union Horizon 2020 Research and Innovation Program Grant 780701. The bulk of this work was done while MA was at Royal Holloway, University of London. MP acknowledges support by EPSRC DTP studentship grant EP/T517811/1. The research of YS was supported by EPSRC grant EP/S02087X/1 and EP/W02778X/1. The research of PW was supported by EPSRC grants EP/T001062/1, EP/T026715/1 and by the ISCF grant 10001712.

\(^{11}\)https://github.com/Milos9304/Latti1Q, https://github.com/Milos9304/Latti1Q
References


Section 3.1 shows that an Ising spin Hamiltonian whose eigenstates correspond to lattice vectors in a certain region can be constructed in a straightforward manner. However this implicitly encodes the zero vector as the ground state of the Hamiltonian. Since the output of the SVP problem is expected to be non-zero, a technique to essentially find the Hamiltonian’s first excited state must be utilised. Note that this is equivalent to finding a new Hamiltonian that has as ground state the first excited state of our initial Hamiltonian. In the case of VQE there exists a natural solution to the problem that can be accommodated fully at the classical loop (classical post-processing) of the optimization. This happens by defining a cost function that excludes the contribution of the zero vector eigenstate and consequently guides the optimiser towards the first excited state. However, such modification cannot be done to QAOA nor can be executed on a quantum annealer as the cost function is strictly defined by the optimization strategy itself.

The first approach one might consider is ignoring the \( x \neq 0 \) and proceed with trying to optimise for the ground state. There is a chance that the overlap of the first excited state with final ansatz state (aimed to match closely the ground state/zero-vector) is non-zero, a technique to essentially find the Hamiltonian’s first excited state must be utilised. Note that this is equivalent to finding a new Hamiltonian that has as ground state the first excited state of our initial Hamiltonian. In the case of VQE there exists a natural solution to the problem that can be accommodated fully at the classical loop (classical post-processing) of the optimization. This happens by defining a cost function that excludes the contribution of the zero vector eigenstate and consequently guides the optimiser towards the first excited state. However, such modification cannot be done to QAOA nor can be executed on a quantum annealer as the cost function is strictly defined by the optimization strategy itself.

The first approach one might consider is ignoring the \( x \neq 0 \) and proceed with trying to optimise for the ground state. There is a chance that the overlap of the first excited state with final ansatz state (aimed to match closely the ground state/zero-vector) is non-zero, and that it gets sampled during the final ansatz measurements. This approach does not increase any requirements on quantum resources, however, especially as the number of qubits increases, it is not expected to yield satisfactory results. The difference in cost between the zero vector and the first excited state (the SVP) increases and the probability of obtaining a penalty for reaching the zero vector (ground state of the first excited state while targeting the ground state) decays possibly exponentially fast.

Instead, a more accurate and in the long term, viable solution is to modify the Hamiltonian and impose a penalty for reaching the zero vector (ground state of the first excited state while targeting the ground state) decays possibly exponentially fast.
the “naive” Hamiltonian). Introducing the extra constraint, and ensuring that the new Hamiltonian is still QUBO (so that we can run QAOA or quantum annealing) requires to introduce auxiliary variables and thus requires more qubits. In our case, we succeed to obtain a theoretical guarantee that the optimization result converges to the shortest lattice vector in the limit of optimization parameters for the cost of introducing extra \( n - 2 \) qubits for a lattice with rank \( n \). Such approach hence roughly reduces the abilities of QAOA algorithm or quantum annealer device by a half compared to the VQE algorithm but allows us to benefit from a wider range of quantum variational approaches and hence we include it here for the reference. A trivial solution would be to penalise certain assignment of QUBO binary variables that makes the QUBO formulation zero. As we will argue below, given \( m \) QUBO binary variables, the approach would require to introduce \( m - 2 \) new binary variables. Note that \( m \geq n \) is the number of binary variables in QUBO. If we let \( n \) be rank of a lattice we can see that \( n \geq n \) as shown in Section 3.1 where \( m \approx n \log(2q) \) if each \( x_i \) has the same variable range \( 2a \) assigned. Suppose that instead of the mapping presented in Equation 2 we instead map each integer variable as

\[
|x_i| \leq a \implies x_i = \frac{a + \zeta_i}{(a) + 1} + \sum_{j=0}^{\lfloor \log(a-1) \rfloor} 2^j \tilde{x}_{ij} + (a - 2\lfloor \log(a-1) \rfloor) \tilde{x}_{i, \lfloor \log(a-1) \rfloor}
\]

by introducing new binary variables \( \zeta_i, \omega_i \) and \( \{ \tilde{x}_{ij} \}_{0 \leq j \leq \lfloor \log(a) \rfloor} \) for each \( x_i \). The encoding (5) allows us to encode penalization of zero state with much less qubits. It is easy to observe that for the last two terms of (5) it holds that

\[
0 \leq \sum_{j=0}^{\lfloor \log(a-1) \rfloor} 2^j \tilde{x}_{ij} + (a - 2\lfloor \log(a-1) \rfloor) \tilde{x}_{i, \lfloor \log(a-1) \rfloor} \leq a - 1
\]

and consequently

\[
x_i = 0 \implies \zeta_i = 1
\]

Hence the penalization of the case where all integer variables are penalised for being zero \( \forall x_i = 0 \) is equivalent to penalization of the case where the same number of binary variables are one \( \forall \zeta_i = 1 \). The penalization term then becomes (6)

\[
P \prod \zeta_i
\]

where \( P \) is the penalty value. The term (6) can be encoded as (7) where \( \{ z_i \}_{1 \leq i \leq n} \) are extra auxiliary binary variables with bijective correspondence to \( \{ \zeta_i \}_{0 \leq i \leq n} \).

\[
P \cdot \left( 1 + \sum_{i=1}^{n} z_i \cdot \left( -1 - \zeta_i \right) + \sum_{k=i+1}^{n} \left( -1 - \zeta_k \right) \right)
\]

(7) encodes constraint (6). Let \( x_i = 1 - \zeta_i, \tau_i = -x_i + \sum_{k=i+1}^{n} x_k \) and let \( z_i = 1 \iff \tau_i < 0 \). Then it is easy to see that values of \( z_1, \ldots, z_m \) minimise (7). If \( \forall x_i = 0 \), the whole expression is equal to \( P \) as required. Otherwise, let \( j_1, \ldots, j_m \) be positions of \( m \leq n \) binary ones in the bitstring \( x_1, x_2, \ldots, x_n \) sorted in the increasing order. Then \( \tau_{j_m} = -1 \) and \( \tau_{j_l} \geq 0 \) for \( l < m \). Hence \( \forall i \neq j_m, z_i = 0, z_{j_m} = 1 \) and the whole expression is 0, hence no penalty is imposed.

Observe that setting \( n \) and \( z_{n-1} = \zeta_n \) does not change the global minimum of (7). Hence \( n - 2 \) additional binary variables \( \{ z_i \}_{1 \leq i \leq n-2} \) are needed to encode the penalization term for an \( n \)-rank lattice. The quadratic unconstrained binary optimization problem formulation that can be trivially mapped to Ising spin Hamiltonian hence becomes

\[
\min_{x_1, \ldots, x_n} \sum_{i=1}^{n} x_i \cdot G_{ij} + 2 \sum_{1 \leq i < j \leq n} x_i \cdot x_j \cdot G_{ij} + L \cdot \left( \sum_{i=1}^{n} z_i \cdot \left( -1 - \zeta_i \right) + \sum_{k=i+1}^{n} \left( -1 - \zeta_k \right) \right)
\]

where \( x_i \) is encoded as in Equation (5). Number of binary variables needed to represent \( x_i \) bounded by \( a_i \) is \( \lfloor \log(a_i) \rfloor + 3 \). Hence the total number of binary variables in Equation (8) is

\[
n_{\text{bin vars}, \text{constrained}} = \sum_{i=1}^{n} \left( \lfloor \log(a_i) \rfloor \right) + (n-2) = 4n - 2 + \sum_{i=1}^{n} \lfloor \log(a_i) \rfloor.
\]

**B Probability of including the shortest lattice vector in the search space using naive qubit mapping strategies**

As discussed in Sections 4 and 5.1, the bounds determined in Section 4.1 are not the best choice for small lattice instances. Given \( n \) dimensional lattice \( \mathcal{L} \) and \( m \) available qubits, choosing a uniform distribution of qubits over elements of the coefficient vector \( x \) turns out to achieve a high probability of encoding the shortest non-zero lattice vector as the ground state of a problem Hamiltonian when \( m \) is sufficiently large. Figure 2 depicts the probabilities of encoding the shortest non-zero lattice vector as a Hamiltonian’s ground state given the number of available qubits and the lattice rank. The results were averaged over 1024 instances prepared using the same methodology as for the quantum emulation experiments described in Section 5.1. Three approaches have been compared:

- **blue** distributing \( \left\lfloor \frac{8n}{m} \right\rfloor \) qubits to each of the coefficient of \( x \),
randomly distributing \( \lfloor \frac{m}{n} \rfloor \) qubits to \( n \times \lfloor \frac{m}{n} \rfloor \), coefficients of \( x \) and distributing \( \lfloor \frac{m}{n} \rfloor + 1 \) to the remaining coefficients of \( x \).

Using Lemma 1 with \( A = gh(L) \) to obtain a discrete distribution of the bounds over coefficients of \( x \); the distribution is then scaled such that at most \( m \) qubits were distributed.

## Conditional value at risk experiment data

We chose the value of \( \alpha = 0.175 \) to use in the \( CVaR_{\alpha} \) VQE algorithm in our quantum emulation experiments in Section 5. Based on an experiment involving 1024 instances of rank 16 lattices prepared as in Section 5.1 this value gave the best results. The value of \( \alpha \) was varied incrementally in small steps. Figures 3a and 3b depict the mean and the median overlaps of the final ansatz state with Hamiltonian’s ground state respectively. We can observe that despite positive linear correlation between the mean overlap and parameter \( \alpha \), the median overlap peaks sharply at \( \alpha \approx 0.175 \), an indication of a rightly-skewed distribution. In other words, we have seen significantly less instances with high overlap as \( \alpha > \approx 0.4 \) tends towards one, although when such overlaps happen, they are high enough so that their mean increases with \( \alpha \). The inverse of the overlap is the expected number of measurement samples of the final ansatz state to obtain the ground state. Hence a compromise must have been made to achieve as many as possible overlaps which were sufficiently high in the sense that the ground state could be obtained from them by a reasonable number of measurements. In order to make a fair comparison, we considered the probabilities of sampling the ground state if 5000 measurements of the final ansatz were to be performed as depicted in Figure 3c. We chose the value \( \alpha = 0.175 \) as it achieved around 78% probability of finding the Hamiltonian’s ground state and at the same time it achieves the highest median of overlaps (around 0.006). Note that the non-\( CVaR_{\alpha} \) VQE version, i.e. when \( \alpha = 1 \) resulted in only 21% probability of finding the Hamiltonian’s ground state.

A final thing to note is that when one uses \( CVaR \), the optimiser has incentive to find a quantum state that has \( \alpha \) overlap with the true ground state, but no incentive to increase the overlap to higher values than \( \alpha \), since anything above the \( \alpha \)-tail is irrelevant for computing the cost. Therefore, while it has been demonstrated that small values of \( \alpha \) may increase the chances of finding the solution after a fixed number of shots/measurement of the final state is carried out, the actual (mean) overlap does not necessarily increase (at least not to values above \( \alpha \)). This was one of the motivation to look modify \( CVaR \) and define an Ascending-C\( CVaR \) [26].

### Proof of Corollary 2

Let \( \delta \) be the orthogonality defect. Then

\[
\log_2 \delta \leq \frac{n}{2} \log_2 \gamma_n + \frac{1}{2} \log_2 \left( \frac{n+3}{n} \right) - n \\
\leq \frac{n}{2} \log_2 \left( \frac{1}{2} n + \frac{9}{8} \right) - n + \frac{1}{2} \log_2 (n+3)! + O(1) \\
= \frac{n}{2} \log_2 (n) - \frac{1}{2} n + \frac{1}{2} \log_2(n+3)! + O(1) \\
= \frac{n}{2} \log_2 (n) - \frac{1}{2} n + \frac{1}{2} \log_2 n! + O(\log n) \\
\leq \frac{n}{2} \log_2 (n) - \frac{1}{2} n + \frac{1}{2} \log_2 \left( \sqrt{2\pi n} \left( \frac{n}{e} \right)^n e^{1/12 n} \right) + O(\log n) \\
= \frac{n}{2} \log_2 (n) - \frac{1}{2} n + \frac{1}{2} \log_2 n - \frac{1}{2} \log_2 e + O(\log n)
\]

where we used [41] at the second to last line.
Figure 2: Probability of encoding the shortest lattice vector as an eigenvector of problem Hamiltonian using blue: uniform distribution of qubits, red: uniform distribution of qubits and random distribution of the remaining qubits, green: scaled distribution of qubits according to Lemma 1 with $A$ being a gaussian heuristics corresponding to the lattice instance.
Figure 3: Experimental determination of CVaR’s parameter $\alpha$

(a) Averaged overlap with the shortest lattice vector as a function of parameter $\alpha$ with standard deviations.

(b) Median overlap with the shortest lattice vector as a function of parameter $\alpha$.

(c) Probability of sampling the ground state in 5000 samples using CVaR version of VQE as a function of parameter $\alpha$. Note that $\alpha = 1$ corresponds to the standard non-CVaR version of VQE.