qLUE: A Quantum Clustering Algorithm for Multi-Dimensional Datasets

Dhruv Gopalakrishnan[1,2,3], Luca Dellantonio[1,4,5], Antonio Di Pilato[6], Wahid Redjeb[6,7], Felice Pantaleo[6], and Michele Mosca[1,3,4,8]

[1]Institute for Quantum Computing, University of Waterloo, Waterloo, ON N2L 3G1, Canada
[2]Department of Electrical and Computer Engineering, University of Waterloo, Waterloo, ON N2L 3G1, Canada
[3]Perimeter Institute of Theoretical Physics, Waterloo, ON N2L 2Y5, Canada
[4]Department of Physics & Astronomy, University of Waterloo, Waterloo, ON N2L 3G1, Canada
[5]Department of Physics and Astronomy, University of Exeter, Stocker Road, Exeter EX4 4QL, United Kingdom
[6]CERN, Geneva
[7]RWTH Aachen University Physikalisches Institut III A, Aachen, Germany
[8]Department of Combinatorics and Optimization, University of Waterloo, Waterloo, ON N2L 3G1, Canada

Clustering algorithms are at the basis of several technological applications, and are fueling the development of rapidly evolving fields such as machine learning. In the recent past, however, it has become apparent that they face challenges stemming from datasets that span more spatial dimensions. In fact, the best-performing clustering algorithms scale linearly in the number of points, but quadratically with respect to the local density of points. In this work, we introduce qLUE, a quantum clustering algorithm that scales linearly in both the number of points and their density. qLUE is inspired by CLUE, an algorithm developed to address the challenging time and memory budgets of Event Reconstruction (ER) in future High-Energy Physics experiments. As such, qLUE marries decades of development with the quadratic speedup provided by quantum computers. We numerically test qLUE in several scenarios, demonstrating its effectiveness and proving it to be a promising route to handle complex data analysis tasks – especially in high-dimensional datasets with high densities of points. The code we used for these simulations is available at Ref. [1]

Dhruv Gopalakrishnan: dgopalak@uwaterloo.ca
Luca Dellantonio: l.dellantonio@exeter.ac.uk
Antonio Di Pilato: tony.dipilato@cern.ch
Wahid Redjeb: wahid.redjeb@cern.ch
Felice Pantaleo: felice.pantaleo@cern.ch
Michele Mosca: michele.mosca@uwaterloo.ca

1 Introduction

Clustering is a data analysis technique that is crucial in several fields, owing to its ability to uncover hidden patterns and structures within large datasets. It is essential for simplifying complex data, improving data organization, and enhancing decision-making processes [2–5]. For instance, clustering has been applied in marketing [6, 7], where it helps segment customers for targeted advertising [8], and in biology, for classifying genes and identifying protein interactions [9–12]. In the realm of computer science and artificial intelligence, it is invaluable for image [13] and speech recognition [14, 15], as well as recommendation systems [16, 17] that personalize content for users. Finally, clustering techniques are pivotal for Event Reconstruction (ER), where data points that originated from the same "event" must be collected together. In High-Energy Physics, for instance, clustering algorithms reconstruct the trajectories of subatomic particles in collider experiments. It is expected that the endcap high granularity calorimeter (HGCAL) [18] being built for the CMS detector at the High Luminosity Large Hadron Collider will provide extremely large volumes of data that must be tackled by new generations of clustering algorithms such as CLUE. The discovery of the Higgs boson [19], awarded the Nobel prize in 2012, was made possible by such algorithms.

ER enables the interpretation of data obtained from particle collision events, including those occurring at the Large Hadron Collider (LHC) at CERN. Several clustering algorithms like DBScan, K-Means, and Hierarchical Clustering among others [20–22] can be employed for ER. Our work is based on CERN's CLUstering of Energy (CLUE) algorithm [23], which is adopted by the CMS collaboration [24–26]. It is designed for the future HGCAL detector due to the limitations of the currently employed algorithms. Despite these limitations, such algorithms are already at the basis of several discoveries, such as the doubly charged tetraquark [27], the observation of four-top quark production in proton-proton collisions [24] and the study of rare B meson decays to two muons [25].

The efficiency of clustering algorithms, as illustrated by the CLUE algorithm [23], is crucial for handling extensive datasets. Initially designed for two-dimensional datasets, CLUE reduces the search complexity from $O(n^2)$ to $O(mn)$ through the use of local density and a tiling procedure, where n (m) represents the (average) number of points (per tile).

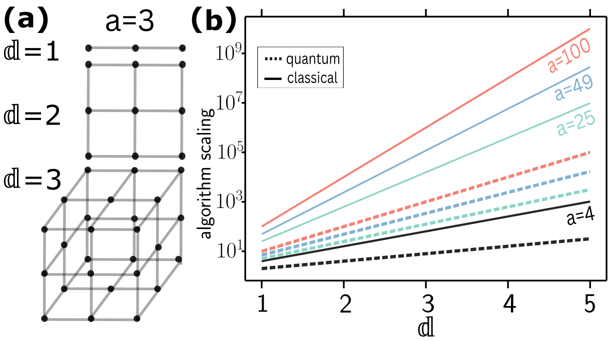

Figure 1: Scaling of point density and complexities of classical and quantum algorithms for the unstructured search problem with dimension $\mathbb{d}$. In (a), different $\mathbb{d}$-dimensional lattices for $\mathbb{d} = 1, 2, 3$ and $a = 3$ points per edge. In (b), best-known classical (solid lines) and quantum (dashed lines) algorithmic scalings for the Unstructured Search Problem [28] applied to square $\mathbb{d}$-dimensional lattices with the values of a reported in the plot. Classically, the cost $O(m)$ reflects the need to iterate through all the m points to find the desired one. Grover achieves the same in $O(\sqrt{m})$ steps, providing a quadratic advantage. This advantage increases with the density of points in the considered dataset, which grows exponentially with respect to the dimension $\mathbb{d}$ according to $m = a^{\mathbb{d}}$.

In the context of CLUE, where data sets are limited to two dimensions, m is small, making this approach to ER particularly effective. However, as the dimensionality of the dataset is incremented, the value of m generally increases exponentially. This is highlighted by Fig. 1(a), where for a $\mathbb{d}$-dimensional lattice with a points per edge, m follows the relation $m = a^{\mathbb{d}}$. This is a serious challenge to CLUE and classical clustering algorithms in general.

A first step towards extending CLUE to more dimensions is done by 3D-CLUE [23, 29]. In this work, data points from different layers of detectors are first projected onto a single $\mathbb{d} = 2$ surface, where clustering is then performed. However, this projection from the original $\mathbb{d} = 3$ dataset to a $\mathbb{d} = 2$ surface comes at the cost of a slower algorithm since m becomes effectively larger. The solid lines in Fig. 1(b) show the increase in average points per tile in $\mathbb{d}$-dimensional datasets made of the lattices in panel (a). While the improved performance of 3D-CLUE in ER tasks [23, 29] justifies the increased computational overhead, extending this enhancement to higher dimensions and larger datasets is challenging. Finding practical approaches to deal with datasets where $\mathbb{d}$ is large is therefore extremely important, not only for ER tasks but also in other fields such as gene analysis [30] and market segmentation in business [31].

Quantum computers provide a route to mitigate the complexity blowup arising from higher dimensional datasets. Ref. [32] addresses the task of jet clustering in High-Energy Physics, while Ref. [33] targets spectral clustering, which itself uses the efficient quantum analog of k-means clustering [34]. Other approaches include quantum k-medians clustering [35] and a quantum algorithm for density peak clustering [36].

In this work we develop qLUE, a CLUE-inspired quantum algorithm. Similarly to other quantum algorithms [37, 38], qLUE leverages the advantage provided by Grover Search [28]. A comparison of classical and quantum (Grover) runtimes is presented in Fig. 1(b), where the solid [dashed] lines refer to the classical $O(m)$ [quantum $O(\sqrt{m})$] scaling. As can be seen, the complexity advantage that Grover search provides can be substantial, particularly for large values of $\mathbb{d}$ or a.

Overall, we find that qLUE performs well in a wide range of scenarios. With ER-inspired datasets as a specific example, we demonstrate that clusters are correctly reconstructed in typical experimental settings. Similar to other quantum approaches to clustering that rely on Grover Search [35, 39], qLUE also showcases a quadratic speedup compared to classical algorithms. The specific advantages of qLUE are its CLUE-inspired approach to cluster reconstruction (which demonstrated to be extremely successful [24, 25, 40]), and its consequent seamless integration with the classical framework currently employed by the CMS collaboration [23, 29, 41].

This paper is structured as follows. In Sec. 2, we describe our algorithm qLUE. Specifically, we provide a general overview of its subroutines – namely the Compute Local Density, Find Nearest Higher, and the Find Seeds, Outliers and Assign Clusters steps. We describe the results of our simulated version of qLUE on a classical computer in Sec. 3. In more detail, we explain the scoring metrics we use to quantify our results, and describe qLUE performance when the dataset is subject to noise and different clusters overlap. Conclusions and outlook are finally presented in Sec. 4.

2 qLUE

qLUE is a quantum adaptation of CERN's CLUE and 3D-CLUE algorithms [23, 29], that is specifically developed for ER, yet it is suitable to work with any (high dimensional) dataset. The main advantage of qLUE stems from employing Grover's algorithm, which provides a quadratic speedup for the Unstructured Search Problem [28]. While qLUE is designed to work in arbitrary dimensions, for clarity we restrict ourselves to $\mathbb{d} = 2$. This simplifies the following discussions and allows us to simulate qLUE with meaningful datasets on a classical computer. Generalizations to higher dimensions can be done following the steps outlined below. Furthermore, to provide a better connection with CLUE and 3D-CLUE, we employ a similar notation.

In Sec. 2.1, we offer an overview of the algorithm and its different subroutines. Sec. 2.2 is dedicated to the first subroutine of qLUE, namely, calculating the Local Density. We then explain how to determine the Nearest Highers ($\mathcal{N}_j$), Seeds, and Outliers in Sec. 2.3. Finally, Sec. 2.4 delves into the conclusive Cluster Assignment subroutine, where the points in the dataset are effectively heirarchically clustered.

2.1 Overview and Setting

As for CLUE and 3D-CLUE [23, 29], we consider a dataset with spatial coordinates and an energy for every point. Similar datasets can also be found in medical image analysis and segmentation [42, 43], in the analysis of asteroid reflectance spectra and hyperspectral astronomical imagery in astrophysics [44–46] and in gene analysis in bioinformatics [30, 47].

In $\mathbb{d} = 2$ dimensions, the spatial coordinates X_j for point j are $X_j = [x_{j,1}, x_{j,2}]$, that are promptly generalized for larger values of $\mathbb{d}$. Both CLUE and qLUE first perform tiling over the dataset to reduce the search and therefore enhance the efficiency of the algorithm. Tiling is the process of partitioning the dataset into a grid of rectangular tiles $\square_k$, where k is the tile index (see Fig. 2). Therefore, our input dataset comprises of point and tile indices j and k, respectively, the coordinates X_j, and a parameter E_j associated to each point. Following CLUE's notation, we call E_j the energy, yet this should be considered as a label that can be employed to improve the clustering quality for any given dataset. The tiling procedure of qLUE and CLUE enables searching only over Search Spaces $\mathcal{S}$ marked by the tiles in green in Fig. 2(a) as opposed to the full dataset. In case of CLUE, this allowed for an improvement in scaling from $O(n^2)$ to $O(mn)$. The scaling of qLUE is investigated below.

In this work, we employ a qRAM to store and access data, which is an essential building block for quantum computers. Following Ref. [48], we therefore assume that we can efficiently prepare the state

$$\sum_j |j\rangle \xrightarrow{\text{qRAM}} |j\rangle\,|D_j\rangle\,, \tag{1}$$

where D_j is the data associated with a given index j, e.g. the j^{th} point in the database. For convenience, here and throughout this paper we do not explicitly write the normalization factors of quantum states.

The qLUE algorithm consists of the following steps:
Local Density: The first step is to calculate the

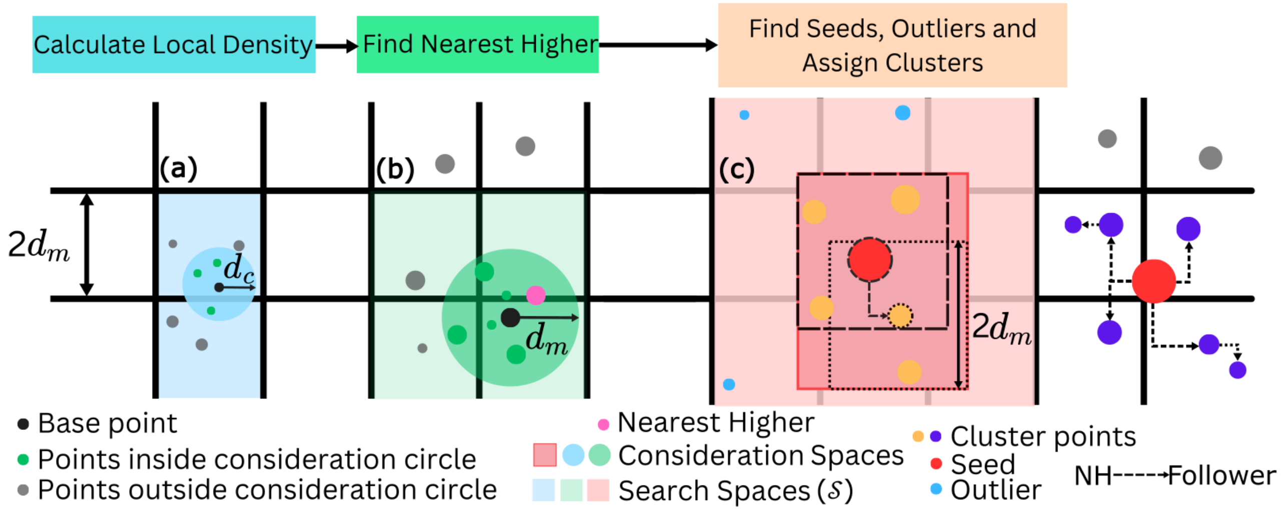

Figure 2: Pictorial representation of the main subroutines of qLUE. In (a), the Local Density computation subroutine is represented. The consideration circle of radius d_{c} (light blue) centered at the base point j (black) contains all points (green) that satisfy $d_{i,j} \leq d_{\mathrm{c}}$. This consideration circle intersects 2 tiles $\square_k$ (indexed by tile index k), highlighted in blue, that form the search space $\mathcal{S}$. As per Eq. (2), the Local Density computation step determines the set of green points from all points in the search space (green and grey) and then computes the local density. In (b), we pictorially present the Find Nearest Higher ($\mathcal{N}_j$) subroutine. The consideration circle (green) around base point j (black) has radius d_{m}. This consideration circle, containing the green points as well as the Nearest Higher $\mathcal{N}_j$ (pink), intersects the 4 tiles highlighted in green, which form the search space $\mathcal{S}$. In (c), we describe the Find Seeds, Outliers and Assign Clusters subroutines. The seeds (red) and outliers (blue) are determined via Grover search on the dataset. In this specific example there are two clusters in the dataset whose non-seed points are in orange and purple, respectively. Followers (see main text) in these clusters are connected by dashed arrows. The Cluster Assignment subroutine is shown to be working on the orange cluster where the cluster $\mathcal{C}$ currently consists of the seed (red, dashed border) and the first of its followers (orange, dotted border). Followers are being found within the Dynamic Search Space (DSS, light red box with solid red border). The DSS is formed as the set of tiles $\square_k$ covered partially or fully by the minimum bounding box of the square windows that contains all the search spaces $\mathcal{S}$ of the points within $\mathcal{C}$.

local density ρ_j of all points j [e.g, black point in Fig. 2(a)] that is defined by

$$\rho_j = E_j + \frac{1}{2}\sum_{d_{i,j}<d_c} E_i \tag{2}$$

and it is indicative of the energy in a neighbourhood of point j. As can be seen from Eq. (2) and Fig. 2(a), ρ_j is a weighted sum over the energies E_i of all points i whose distance $d_{i,j} = \sqrt{\sum_{\alpha=1}^{\mathrm{d}}(x_{i,\alpha} - x_{j,\alpha})^2}$ from the base point j is within a user-specified critical radius d_{c} that characterizes the consideration circle for the Local Density computation subroutine (light blue circle in the figure). As such, E_i is the energy of the i^{th} point which is $d_{i,j}$ away from point j. The choice of weight 1/2 for E_j in the definition of ρ_j in Eq. (2) is empirically found to yield better performances for CLUE [23].

Find Nearest Higher: After calculating the local densities, we determine the nearest highers. The Nearest Higher $\mathcal{N}_j$ of a point j is the point nearest to j with a higher local density $\rho_{\mathcal{N}_j} > \rho_j$. As better explained in Sec. 2.4, the Nearest Higher are used to heirarchically cluster points together in the Cluster Assignment process at the end of qLUE. In Fig. 2(b), the Nearest Higher $\mathcal{N}_j$ of the base point j (black point) is the pink point.

Find Seeds, Outliers and Assign Clusters: As schematically represented in Fig. 2(c), seeds (red points) are the points whose distance $d_{j,\mathcal{N}_j}$ from their Nearest Higher $\mathcal{N}_j$ and whose local density ρ_j are lower bounded by user defined thresholds. Outliers (blue points) are the points whose distance from Nearest Higher is similarly lower bounded but whose Local Density has an upper threshold. As such a point j is

$$\text{a } \textit{seed} \text{ if } d_{\mathcal{N}_j,j} > d_{\mathrm{c}} \text{ and } \rho_j > \tilde{\rho}, \tag{3a}$$

$$\text{an } \textit{outlier} \text{ if } d_{\mathcal{N}_j,j} > \delta d_{\mathrm{c}} \text{ and } \rho_j < \tilde{\rho}. \tag{3b}$$

Here, δ is the Outlier Delta Factor that determines the upper bound on the allowed local density for outliers. Furthermore, $\tilde{\rho}$ is the critical density threshold – the lowest local density a point can have to be classified as a seed. Both

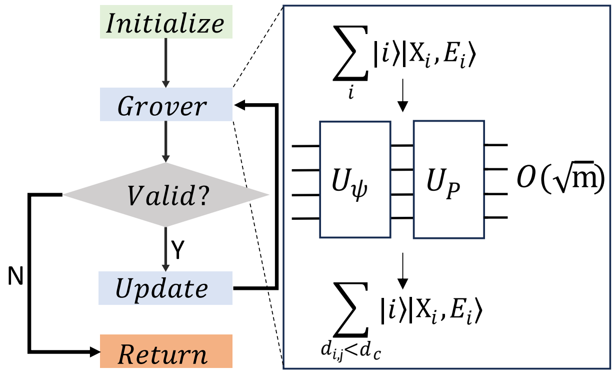
Initialize
Grover
Valid?
Y
N
Update
Return
$\sum_i |i\rangle|\mathrm{X}_i, E_i\rangle$
U_ψ
U_P
$O(\sqrt{\mathrm{m}})$
$\sum_{d_{i,j}<d_c} |i\rangle|\mathrm{X}_i, E_i\rangle$

Figure 3: Algorithm flow for Local Density computation and for Assigning Clusters. The quantum state is initialized in the green "Initialize" box. For Local Density Computation (Cluster Assignment), it comprises all points in the DSS $\mathcal{S}$ (in the DSS). The "Grover" (light blue) block performs U_ψ and U_P in succession $O(\sqrt{m})$ times, and returns all points satisfying the required condition. The inset considers the case of Local Density computation where the condition is $d_{i,j} < d_c$. For the cluster assignment step, we check if points in the DSS are followers of the points in the cluster $\mathcal{C}$ (see Sec. 2.4). The output of the Grover subroutine is then measured to yield an index that is checked for validity in the grey "Valid?" diamond. If the point satisfies the chosen condition, the Y branch is executed. Within the "Update" (light blue) step this point is then removed from either $\mathcal{S}$ or the DSS and stored to be returned in the "Return" orange box. Once all points are found, the "Valid?" condition triggers the N branch to terminate the algorithm. Depending on the chosen subroutine, the returned indices are employed to compute the Local Density from Eq. (2), or to construct $\mathcal{C}$.

δ and $\tilde{\rho}$ are user-specified and can be varied to enhance the quality of the output. Seeds are generally located in areas of high energy density, and will be employed as starting points to build clusters. Outliers are points that are likely to be noise in the dataset and are therefore discarded.

Once seeds and outliers are determined, the clusters are constructed. From the seeds, we iteratively combine "followers". If point $\mathcal{N}_j$ is the Nearest Higher of point j, then point j is termed as $\mathcal{N}_j$'s follower. The follower of a point is most likely generated by the same process as the point itself (in the context of ER, by the same particle), and as such shall be included in the same cluster. In Fig. 2(c), the orange and purple points form two different clusters, and the followers of the points in the purple one are indicated by arrows.

2.2 Local Density Computation

In this section, we describe the subroutine (schematically represented in Fig. 3) that computes the Local Density ρ_j of the point j, as defined in Eq. (2). To perform the computation, all points i whose distance $d_{i,j}$ from point j is smaller than the threshold d_c need to be determined from the search space $\mathcal{S}$. This search space is the smallest set of tiles $\square_k$ required to cover the $d_{i,j} < d_c$ consideration circle. In Fig. 2(a), $\mathcal{S}$ is highlighted in light blue.

We shall refer to $\mathcal{S}$ as the local dataset that, as explained above, can be efficiently prepared with the qRAM [48]. To do so, we only require determining the tiles $\square_k$ that are in the search space, which can be done efficiently classically [23]. The initial state of this subroutine, after being prepared via the qRAM, is therefore

$$\sum_{k\in\mathcal{S}}\sum_{i\in\square_k}|i\rangle \xrightarrow{\text{qRAM}} \sum_{k\in\mathcal{S}}\sum_{i\in\square_k}|i\rangle\,|X_i,E_i\rangle\,, \tag{4}$$

where the index i is unique for each point in $\mathcal{S}$. $i \in \square_k$ indicate all indices within tile k [either of the light blue squares in Fig. 2(a)]. Ancillary qubits, omitted for clarity in Eq. (4), are employed within the Grover search (for more information, see App. A).

At this stage, we must find the points i [green dots in Fig. 2(a)] that are within a radius of d_c from the base point j [black point in Fig. 2(a)]. As shown in Fig. 3, we perform Grover Search to prepare [49]

$$\sum_i |i\rangle\,|X_i,E_i\rangle \xrightarrow{\text{Grover}} \sum_{d_{i,j}<d_c} |i\rangle\,|X_i,E_i\rangle\,. \tag{5}$$

Here, the first register of the Grover output contains all points characterized by indices i such that $d_{i,j} < d_c$. As shown in the inset of the figure, the Grover Search consists of $O(\sqrt{m})$ repetitions (where m is the number of points in $\mathcal{S}$) of the U_ψ and U_P operators. U_P is the diffusion operator and U_ψ is the unitary associated with the oracle of Grover Search [28]. Further details regarding Grover Search and the unitaries we use for our algorithm can be found in App. A.

When the algorithm is run, measurement either yields a point that satisfies this distance condition, or (if there are no valid indices left) an index that does not satisfy this condition. This is verified by the grey "Valid?" diamond in Fig. 3.

The branched logic following this block ensures that the algorithm loops until all the required points are returned by the algorithm in the "Return" block.

Once we have obtained all indices i of points satisfying the distance condition ($d_{i,j} < d_{\mathrm{c}}$), we perform the summation in Eq. (2). This is computed and stored in the original dataset for each point. The database is now updated using qRAM with local density values for all points where the j^{th} point in the database has the corresponding computed local density ρ_j.

The scaling of the subroutine that determines the local density of a single point is given by the number of points in the blue consideration circle in Fig. 2(a) such that $d_{i,j} < d_{\mathrm{c}}$. If we say this number is p, $O(p)$ runs are required. This is therefore a $O(p\sqrt{m})$ algorithm as opposed to the $O(m)$ classical iterative algorithm for the Unstructured Search Problem.

As a final remark, we highlight that it is in principle possible to design a unitary that computes the Local Density directly and stores the output in a quantum register. This unitary would remove the requirement of finding individually the indices i such that $d_{i,j} < d_{\mathrm{c}}$, thus removing the overhead of p in $O(p\sqrt{m})$. However, designing this circuit is non-trivial and its depth may be large. This is therefore left for future investigations.

2.3 Find Nearest Higher

Here, we describe qLUE's subroutine for finding the Nearest Highers ($\mathcal{N}_j$) introduced in Sec. 2.1. As a reminder, $\mathcal{N}_j$ is the nearest point to the base point j whose local density $\rho_{\mathcal{N}_j}$ is more than the local density ρ_j of the base point, see Eq. (3a).

Similar to the initialization carried out for the Local Density Computation step, we use qRAM to initialize the quantum state

$$\sum_{k\in\mathcal{S}}\sum_{i\in\square_k}|i\rangle \xrightarrow{\text{qRAM}} \sum_{k\in\mathcal{S}}\sum_{i\in\square_k}|i\rangle\,|X_i\rangle\,|\rho_i\rangle\,. \qquad (6)$$

Here, the indices i are within the tiles $\square_k$, as in Eq. (4), and $\mathcal{S}$ is the considered search space, schematically represented by the light green box in Fig. 2(b). This search space is determined from d_{m} as opposed to d_{c}, which is the user-defined threshold that is set to be δd_{c}. Note that the energy E_i, employed for determining the densities ρ_i in Sec. 2.2, is hereon not required.

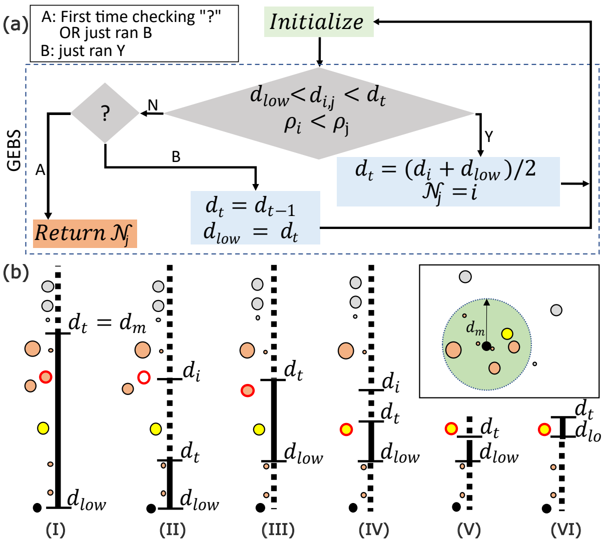

Figure 4: (a) Diagrammatic representation of the algorithm. GEBS determines successive candidates for the "Nearest Higher" until the proper one is found. The quantum state in Eq. (6) is prepared in the "Initialize" step (green box). Grover Search (larger diamond) is then performed to find the points satisfying $d_{\mathrm{L}} < d_{i,j} < d_{\mathrm{t}}, \rho_i < \rho_j$. If this condition is satisfied ('Y' branch), d_{t} is updated and Grover run again. If not ('N' branch), control flows to the "?" diamond. The branch A is entered if the "?" condition is being checked for the first time or if branch B was just run. Branch B is entered if branch Y was just run. (b) The algorithm's working is shown step-by-step (numbers at the bottom) for the search space $\mathcal{S}$ in the inset in the top right corner. The points are mapped to a line where the height represents the distance $d_{i,j}$ from the base point j (black dot at the bottom). The grey (orange) points are outside (inside) the green consideration circle with radius d_{m} [see also Fig. 2(b)]. At each step of GEBS, the thresholds d_{L} and d_{t} are updated according to the logic in panel (a). The dot with the red border indicates the current candidate for $\mathcal{N}_j$; when filled (empty) it is (not) found by Grover Search at that step. The yellow point is the Nearest Higher $\mathcal{N}_|$ that is found at the end of GEBS.

To find the Nearest Higher, we use a Grover-Enhanced Binary Search (GEBS) where each search step is enhanced by Grover's algorithm. The output of every Grover run,

$$\sum_{\substack{d_{\mathrm{L}}<d_{i,j}<d_{\mathrm{t}},\\ \rho_i>\rho_b}} |i\rangle\,|X_i,\rho_i\rangle\,, \qquad (7)$$

is a superposition over all points i whose distance $d_{i,j}$ from the base point j lies between the thresholds d_{L} and d_{t}. Furthermore, their local density ρ_i should be higher than that of the base ρ_j. At each step, d_{L} and d_{t} are updated based on whether a point satisfying the conditions in the grey diamond of Fig. 4(a) is found. Ancilla registers are used here as detailed in App. A.

To better understand the algorithm, we provide a step-by-step walkthrough of the example in Fig. 4(b). The search space $\mathcal{S}$ is schematically represented by the inset in the right hand side, where each dot represents a point with a size that is proportional to its local density. The consideration circle (light green, dotted border) highlights all points within a radius $d_{\mathrm{m}} = \delta d_{\mathrm{c}}$. In this work, we set the outlier delta factor δ to 2. The consideration circle in the inset corresponds to $d_L = 0$ and $d_{\mathrm{t}} = d_{\mathrm{m}}$, shown in step (I). In the main panel, vertical lines refers to the steps (I-VI) of GEBS that are reported below, and schematically represent the distances of all points (coloured dots) from the base point j (black one at the bottom).

GEBS starts with the higher threshold set as $d_{\mathrm{t}} = d_{\mathrm{m}}$ and the lower threshold $d_{\mathrm{L}} = 0$ as shown in vertical line (I) of Fig. 4(b). Following the probabilistic nature of quantum mechanics, assume that the point with a red border indexed i is found after measuring the output of the Grover Search in Eq. (7). This triggers the updates in the Y branch in the diagram of Fig. 4(a), such that we assign $\mathcal{N}_j = i$ and update $d_{\mathrm{t}} \mapsto (d_{i,j} + d_{\mathrm{L}})/2$. The point indexed i is then removed from the search space, as can be seen in (II). Now, since no point satisfies the conditions in the diamond of the flow diagram [see (II)] and d_{t} was just set to $(d_{i,j} + d_{\mathrm{L}})/2$, the B branch is carried out. This updates the thresholds d_{t} and d_{L} for the next iteration of the algorithm, see (III).

Now, assume that the new point with a red border is found [step (III)]. Updates in the Y branch of Fig. 4(a) are carried out again with a new index i and the search region is reduced to contain a single point. In the next step (IV), that point (yellow) is found and, for the third and last time, the nearest higher and the thresholds are triggered according to the Y branch. Next, since no point is found in (V), qLUE executes the updates in the B branch of the diagram. In the last iteration (VI), no points satisfy the desired conditions. The parameter d_{t} was just set to $d_{\mathrm{t}-1}$, i.e, the subroutine just ran B which means that the A branch is now executed and $\mathcal{N}_j$ is returned.

The runtime complexity of the GEBS procedure, with m points in the search space $\mathcal{S}$, is $O(\alpha\sqrt{m})$ as opposed to $O(m)$ classically. The α term is due to the binary search procedure and depends on the size of the quantum register used to encode the distance. Specifically, for a chosen precision $2^{-\Delta}$ used for the positions of the points in the datasets, $\alpha = \Delta$.

2.4 Find Seeds, Outliers, and Assign Clusters

Once the Nearest Highers $\mathcal{N}_j$ are determined for all points j in the dataset, Seeds and Outliers are found via another Grover Search over all points in the dataset. As per the definition in Eq. (3a), Seeds [red points in Fig. 2(c)] are the points with highest local density within a neighbourhood. Outliers [blue points in Fig. 2(c)] are mathematically described by Eq. (3b), are most likely noise, and therefore do not belong to any cluster.

Similar to the previous subroutines, the quantum registers for these procedures are initialized via qRAM. Seeds and outliers are then determined based on the corresponding conditions via Grover Search. Two quantum registers, the first marking whether a point is an outlier and the second to store the seed number – which is also the cluster number – are added to the quantum database.

The final subroutine of qLUE is the assignment of points to clusters. At this stage, outliers have been removed from the input dataset, as they have been already identified. The algorithm flow is the same as that of the Local Density step in Fig. 3. For a chosen seed s, we define $\mathcal{C}$ to be the set containing the indices of all points determined to be in the associated cluster at the end of this subroutine. To assign points to $\mathcal{C}$, we follow a procedure similar to that of the Local Density step in Fig. 3. In the "Initialize" step, $\mathcal{C}$ is initialized to $\{s\}$ and the quantum registers are initialized via qRAM to the state

$$\sum_{i \in \mathrm{DSS}} |i\rangle \xrightarrow{\mathrm{qRAM}} \sum_i |i\rangle\, |\mathrm{V}_i\rangle\,, \tag{8a}$$

$$|\mathrm{V}_i\rangle = |X_i, \rho_i, d_{\mathcal{N}_i,i}, X_{\mathcal{N}_i}\rangle\,. \tag{8b}$$

In the "Grover" block, we search over a superposition of points in the dataset which we call the Dynamic Search Space (DSS). The DSS differs from the search space $\mathcal{S}$ in the Local Density step as it is dynamic. This is because it depends on the points in $\mathcal{C}$, which are updated at each iteration. In Fig. 2(c), for instance, the red seed and the orange point both with black borders are the elements of the current $\mathcal{C}$. To find the DSS, a square window of edge $2d_{\mathrm{m}}$ is first opened for every point in $\mathcal{C}$ (in the figure, the squares with the same border style as the corresponding points). A rectangular region (red box) is then obtained by finding the axis-aligned minimum bounding box for these windows. The set of tiles $\square_k$ covered partially or fully by this minimum bounding box is the DSS. For example, in Fig. 2(c), it comprises the 9 tiles highlighted in light red.

With a similar procedure as for the Local Density subroutine, the "Grover" block now systematically identifies all followers of all points within set $\mathcal{C}$. Here, in the "Update" step in Fig. 3, as the point found by the "Grover" block has passed the "Valid" condition, it is appended to $\mathcal{C}$. Once no more points are found, the "Return" block yields $\mathcal{C}$, following the same flow as the Local Density computation subroutine.

The complexity of the Cluster Assignment step is similar to the one of the Local Density Computation subroutine. The quantum advantage stems from the quadratic speedup provided by the Grover algorithm, which allows determining the follower faster if compared to CLUE. If there are f points in a cluster $\mathcal{C}$ and m points in the corresponding DSS, the classical complexity of the Cluster Assignment step is $O(m)$, while the quantum algorithm has a runtime of $O(f\sqrt{m})$.

3 Results

In this section, we test qLUE in multiple scenarios, each designed to investigate its performance for different settings. In Sec. 3.1, we introduce the scoring metrics used for our analysis. In Sec. 3.2, we describe the performance of the algorithm applied on a single cluster in a uniform noisy environment. In Sec. 3.3, we study the performance on overlapping clusters. Finally, in Sec. 3.4, we study the performance of qLUE on non-centroidal clusters with and without an energy profile.

3.1 Scoring metrics: Homogeneity and Completeness scores

To weigh more energetic points such as seeds higher than the others, we use modified, energy-aware versions [51] of the Homogeneity ($\mathcal{F}_H$) and Completeness ($\mathcal{F}_C$) scores [52]. These metrics are defined in terms of the predicted cluster labels $\mathcal{C}_{\mathrm{p}}$ obtained from qLUE, and the true cluster labels $\mathcal{C}_{\mathrm{t}}$ of the generated dataset. $\mathcal{F}_H$ and $\mathcal{F}_C$ are based on the energy aware [51] mutual information $I(\mathcal{C}_{\mathrm{p}} : \mathcal{C}_{\mathrm{t}})$, the Shannon entropy $H(\mathcal{C}_{\mathrm{t}})$, and the joint Shannon entropy $H(\mathcal{C}_{\mathrm{t}}, \mathcal{C}_{\mathrm{p}})$ [53]:

$$\mathcal{F}_H = \frac{I\left(\mathcal{C}_{\mathrm{p}} : \mathcal{C}_{\mathrm{t}}\right)}{H\left(\mathcal{C}_{\mathrm{t}}\right)} \text{ and } \mathcal{F}_C = \frac{I\left(\mathcal{C}_{\mathrm{p}} : \mathcal{C}_{\mathrm{t}}\right)}{H\left(\mathcal{C}_{\mathrm{p}}\right)}, \tag{9a}$$

$$H\left(\mathcal{C}_{\mathrm{p}}\right) = -\sum_a \frac{E_a}{E} \log_2 \frac{E_a}{E}, \tag{9b}$$

$$H\left(\mathcal{C}_{\mathrm{t}}\right) = -\sum_b \frac{E_b}{E} \log_2 \frac{E_b}{E}, \tag{9c}$$

$$H\left(\mathcal{C}_{\mathrm{p}}, \mathcal{C}_{\mathrm{t}}\right) = -\sum_a \sum_b \frac{E_{a,b}}{E} \log_2 \frac{E_{a,b}}{E}, \tag{9d}$$

$$I\left(\mathcal{C}_{\mathrm{p}} : \mathcal{C}_{\mathrm{t}}\right) = H\left(\mathcal{C}_{\mathrm{p}}\right) + H\left(\mathcal{C}_{\mathrm{t}}\right) - H\left(\mathcal{C}_{\mathrm{p}}, \mathcal{C}_{\mathrm{t}}\right). \tag{9e}$$

As discussed in [51], E_a is the energy aggregated over all points that qLUE classifies into cluster a. E_b is the energy aggregated over all points in cluster b in the true dataset. $E_{a,b}$ is the sum of energies of all points in cluster b in the true dataset that are also assigned to cluster a by qLUE. E is the accumulated energy of all points in the dataset. We remark that for unit energies, Eqs. (9) reduce to the more common form presented in Ref. [52].

qLUE applied to an input dataset yields homogeneity $\mathcal{F}_H = 1$ if all of the predicted clusters only contain data points that are members of a single true cluster. On the other hand, $\mathcal{F}_C = 1$ is obtained if all the data points that are members of a given true cluster are elements of the same reconstructed cluster. Therefore, these metrics are better suited to different scenarios. The impacts of noise and cluster overlap investigated in Secs. 3.2 and 3.3 are better captured by $\mathcal{F}_H$. Indeed, if qLUE incorrectly classifies noise points into predicted clusters, $\mathcal{F}_C$ is unaffected. On the other hand, $\mathcal{F}_C$ shall be employed when studying non-centroidal clusters in Sec. 3.4, since $\mathcal{F}_H = 1$ if one true cluster is divided by qLUE into many sub-clusters.

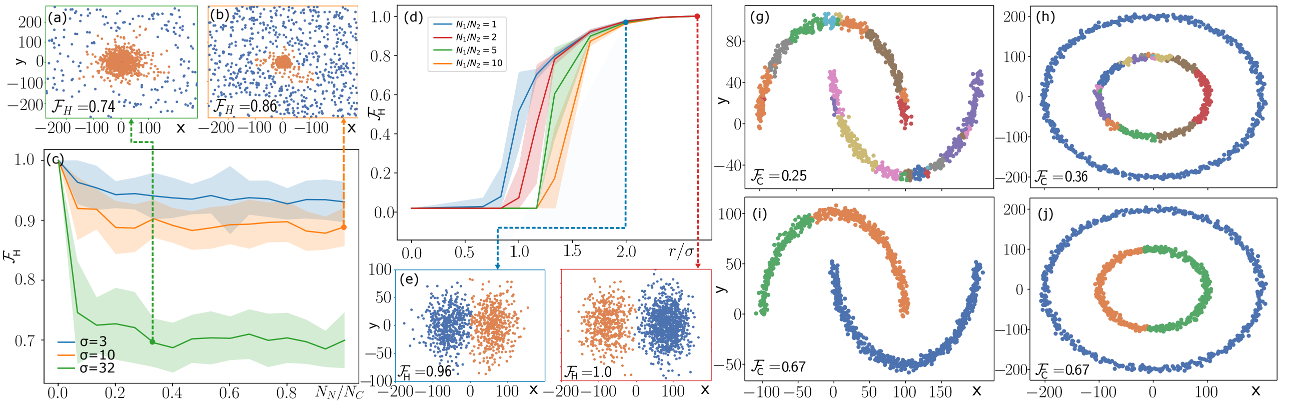

Figure 5: Numerical results from qLUE simulated on a classical machine. (a-c) qLUE's performance in noisy environments. The dataset generated for these experiments and visualized in panels (a) and (b) consists of a cluster (noise) with $N_C = 750$ (N_N) points sampled from the Gaussian distribution in Eq. (10) (uniform distribution) over a square of size 500. The energy of noise points is sampled uniformly between zero and one, while each cluster point is assigned an energy that is the probability of being sampled multiplied by a factor $A = 500$. (a-b) Computed clusters at $N_N/N_C = 0.33, \sigma = 32$, and $N_N/N_C = 1, \sigma = 10$, respectively. In (c), $\mathcal{F}_H$ is plotted against N_N/N_C for the σ in the legend. (d-f) Performance for overlapping clusters. In (d), $\mathcal{F}_H$ vs r/σ is shown for $\sigma = 30$ and different ratios N_1/N_2. Here, r is the distance between the centers of two clusters with $N_1 = 500$ and N_2 points, and we assign to each point an energy that is equal to its sampling probability in Eq. (10). (e-f) Computed clusters at $r/\sigma = 2.0, N_1/N_2 = 1$, and $r/\sigma = 2.67, N_1/N_2 = 2$, respectively. The shadowed regions in (c-d) represent the standard deviations of $\mathcal{F}_H$ over 30 iterations. (g-j) Performance over non-centroidal clusters of 500 points each generated from $scikit-learn$ [50]. In (g-h) the points' energy profile is uniform, while in (i-j) is varied linearly with respect to the distance such that each cluster has a single, most energetic point (see Sec. 3.4). For all experiments, d_c was set to 20 and $\tilde{\rho}$ was set to 25. (a-f) use the energy-aware metric in Eqs. (9) while in (g-j), since the energy profile is assigned by the user and is not part of the dataset itself, in the scoring process we set all points to have the same energy.

3.2 Noise

Here, we study the performance of qLUE for a single cluster in a noisy environment. We vary the number N_N of noise points sampled from a uniform distribution over a square region of fixed size. A cluster of N_C points with coordinates $X_j = [x_{j,1}, x_{j,2}]$ is sampled from the multivariate Gaussian distribution

$$pdf(X_j) = \frac{e^{-\frac{1}{2}(X_j-\mu_j)^T\Sigma^{-1}(X_j-\mu_j)}}{(2\pi)^{\frac{n}{2}}|\Sigma|^{\frac{1}{2}}}, \qquad (10)$$

where $\mu = [\mu_{x_1}, \mu_{x_2}]^T$ is the mean of the distribution (set to $[0,0]^T$ in our case) and Σ the covariance matrix. Here, we choose $\Sigma = \sigma\mathcal{I}$, with $\mathcal{I}$ being the identity matrix and σ a positive real number.

Examples of the generated clusters (in orange) and noise (in blue) are given in Fig. 5(a-b) for $N_N/N_C = 0.33$ at $\sigma = 32$ and $N_N/N_C = 1$ at $\sigma = 10$, respectively. The energy assigned to each point X_j in the cluster is given by $A \times pdf(X_j)$ [see Eq. (10)] with $A = 5 \times 10^2$. The energy of each noise point is randomly sampled between zero and one. This choice resembles the typical scenarios in ER tasks which CLUE [23] was designed for.

In Fig. 5(c), we show the variation of homogeneity score $\mathcal{F}_H$ with respect to the ratio N_N/N_C. We employ the values of σ reported in the legend, that are associated to different colours in the plot. As can be seen, the clustering performance is inversely proportional to both N_N/N_C and σ. When these parameters are small, the typical distance between cluster points is much smaller than that between noise points, and $\mathcal{F}_H$ approaches unity. With a higher chance of labeling noisy points as within the cluster, however, $\mathcal{F}_H$ is lowered. As such, the degrading of $\mathcal{F}_H$ is proportional to the probability of a noise point being in the cluster region, which increases with both σ and N_N/N_C.

3.3 Overlap

Here, we consider the case of two circular clusters with N_1 and N_2 points respectively, each sampled from the multivariate Gaussian distribution in Eq. (10) and with $\Sigma = \sigma\mathcal{I}$. The energy profile is determined by $pdf(X_j)$ for coordinates X_j. The centers μ_1 and μ_2 (two instances of μ) are chosen to be $(r/2, 0)$ and $(-r/2, 0)$, respectively, such that the distance between the cluster centers is r.

In Fig. 5(d), we study the variation of homogeneity score $\mathcal{F}_H$ as a function of r/σ for several values of N_2/N_1. The computed clusters for $r/\sigma = 2$ at $N_2/N_1 = 1$ and $r/\sigma = 2.67$ at $N_2/N_1 = 2$ are shown in panels (e) and (f), respectively, to showcase the typical scenarios considered here.

For all N_1/N_2, $\mathcal{F}_H$ is zero for low r/σ (high overlap). There is then a region where $\mathcal{F}_H$ increases with r/σ and then saturates at unity for high r/σ (little to no overlap). When the two clusters are too nearby, i.e., $r/\sigma \ll 1$, they are in fact indistinguishable and qLUE labels all points together. Increasing the ratio r/σ makes the clusters move away from each other and thus qLUE can discern them. Importantly, large values of $\mathcal{F}_H$ are already attained when the clusters still have a significant overlap. In this scenario, employing the energy labels and the energy density considerably contributes to assigning the points to the right cluster. In fact, the nearest higher points are more likely to connect the points near or on the decision boundary with the more energetic core, thus separating the clusters better.

The performance of qLUE is also affected by the ratio N_1/N_2. When one cluster contains more points than the other, it is more likely to "capture" points from the smaller. The resulting loss in homogeneity score $\mathcal{F}_H$ for low r/σ ratios is evident from Fig. 5(d), where it can be seen that clusters of similar sizes are better distinguished from each other.

3.4 Non-centroidal Clusters

Finally, we study the performance of qLUE on non-centroidal clusters. For this purpose, we use the Moons and Circles datasets in Fig. 5(g-j), generated using $scikit-learn$ [50]. Two settings are considered - one where a uniform energy profile is applied over the points [panels (g-h)] and one where a linear gradient energy profile is employed [panels (i-j)].

In the latter case, for each cluster we assign the highest value of the energy to a single point and lower the energies of all other points proportionally to their x_2 coordinate. In the case of the moon dataset, $E = x_2$ for the upper moon (so the top point of the upper moon has the maximum energy in the cluster) and $E = 60 - x_2$ for the lower moon (so the bottom point has the highest energy in the cluster). For the circles, $E = |x_2 - 200|/10$ for the inner circle and $E = |x_2 + 100|/5$ for the outer one.

Since these datasets have no noise and are well separated, $\mathcal{F}_H$ is always one and we employ $\mathcal{F}_C$ to characterize the performance of qLUE. As in Fig. 5(g-h) the energy profile is uniform, and several points satisfy the seed condition. Therefore, qLUE groups each circle into several clusters, such that we obtain limited values for $\mathcal{F}_C$. On the contrary, cases with an energy profile assigned [Fig. 5(i-j)] results in less seeds that are better recognized by qLUE, and the completeness score $\mathcal{F}_C$ is considerably enhanced.

4 Conclusion and Outlook

We introduced qLUE, a novel quantum clustering algorithm designed to address the computational challenges associated with high-dimensional datasets. The significance of qLUE lies in its potential to efficiently cluster data leveraging quantum computing, mitigating the escalating computational complexity encountered by classical algorithms as dimensions increase. The algorithm's ability to navigate high-dimensional spaces is particularly promising when the density of points is very large, such that local searches become too demanding for classical computers. Therefore, qLUE will be beneficial in multiple scenarios, ranging from quantum-enhanced machine learning [54, 55] to complex data analysis tasks [56].

According to our numerical results, qLUE works well and its performance is significantly enhanced when an energy profile is assigned. Specifically, we study qLUE in noisy environments, on overlapping clusters, and on non-centroidal datasets that are commonly used to benchmark clustering algorithms [57, 58]. In scenarios that are typically encountered in ER tasks, qLUE correctly reconstructs the true clusters to a high level of accuracy. On the other hand, an energy profile can significantly boost qLUE performance in the case of non-centroidal clusters. Our numerical results, backed up by the well-studied

CLUE and by the quadratic speedup stemming from Grover search, make qLUE a promising candidate for addressing high-dimensional clustering problems [32, 33, 36].

As a first outlook, we identify the implementation of qLUE on NISQ hardware [59–65]. This requires a comprehensive consideration of real device constraints. Aspects such as circuit optimization [66], and the impact of noise will be critical and must be carefully addressed. Second, it is possible to improve the scaling of qLUE by devising a unitary that mitigates the need for repeating Grover's algorithm and thereby eliminating the factors of p, α, and f in the scaling of the subroutines outlined in Secs. 2.2, 2.3 and 2.4 respectively. Finally, it is worth investigating variations of qLUE that improve the quality of clustering in different scenarios. For instance, one can devise more sophisticated criteria for the Nearest Higher or Local Density computation steps. One can also improve performance by performing exhaustive hyperparameter searches or via hyperparameter optimization algorithms [67].

Acknowledgements

We thank the CERN Quantum Initiative, Fabio Fracas for creating the fertile ground for starting this project and Andrew J. Jena as well as Priyanka Mukhopadhyay for theoretical support. WR acknowledges the Wolfgang Gentner Programme of the German Federal Ministry of Education and Research (grant no. 13E18CHA). LD acknowledges the EPSRC quantum career development grant EP/W028301/1. DG and MM acknowledge the NTT PHI Lab for funding. Research at IQC is further supported by the Government of Canada through Innovation, Science and Economic Development Canada (ISED). Research at Perimeter Institute is supported in part by the Government of Canada through ISED and by the Province of Ontario through the Ministry of Colleges and Universities.

Appendix

A Grover's Algorithm

Grover's algorithm is a quantum algorithm to solve the Unstructured Search Problem. From a superposition of all states to be searched over, Grover's algorithm involves successive applications of two operators U_ψ and U_P to ensure that the measurement result at the end of the algorithm gives the search output with high probability. We use this algorithm extensively in our work. The inset of Fig. 3 describes the flow of this algorithm for our Local Density computation step. For m points, this involves $O(\sqrt{m})$ successive applications of the operators

$$U_\psi = 2\,|\psi\rangle\langle\psi| - 1, \tag{11a}$$

$$U_P\,|x\rangle = \begin{cases} -\,|x\rangle & f(x) = 1 \\ |x\rangle & f(x) = 0 \end{cases}. \tag{11b}$$

Here, $f(x) = 1$ when the current point satisfies a desired condition (e.g in the context of Local Density Computation, it lies in the critical radius d_c). If this condition is not satisfied, $f(x) = 0$.

To implement the operators in Eqs. (11), we require A(dd) and M(ultiply) circuits. We use the ones introduced in Ref. [68], which perform the following operations

$$\mathrm{A}\,|\mathrm{X_i}\rangle\,|0\rangle = |X_i\rangle\,|x_{1_i} + x_{2_i}\rangle\,, \tag{12a}$$

$$\mathrm{M}\,|\mathrm{X_i}\rangle\,|0\rangle = |X_i\rangle\,|x_{1_i} \cdot x_{2_i}\rangle\,. \tag{12b}$$

For the local density step, the quantum circuit U_P to implement the search function f is given in Fig. 6. The overall idea is to compute the Euclidean distance between every input point and the base point and check if this distance is higher than d_c. The $|sign\rangle$ qubit stores the output of this computation. The F gates are X_{sign} gates which are X on the sign qubit and act as the identity on every other qubit, such that $F\,|x_{1_i}\rangle = |-x_{1_i}\rangle$. These are used such that the first and second levels of Add gates A compute $x_{i_1} - x_{i_2}$ into $|a_i\rangle$ and $|b_i\rangle$ respectively. The M multiply gates then set the $|m_i\rangle$ states to $(x_{i_1} - x_{i_2})^2$ taking a_i and b_i as inputs. An A gate next acts on m_1 and m_2 to set $|ans\rangle$ to $(x_{1_1} - x_{1_2})^2 + (x_{2_1} - x_{2_2})^2$. The A_sign gate is a subcircuit of the addition circuit A that finally computes the sign of $(x_{1_1} - x_{1_2})^2 + (x_{2_1} - x_{2_2})^2 - d_c^2$ and stores it in $|sign\rangle$. Thus, $|sign\rangle = |f(x)\rangle$ [with $f(x)$ as in Eq. (11b)] for the Local Density computation step.

For the Nearest Higher Procedure, a similar circuit can be used with additional registers $|\rho_i\rangle$ for the Local Density and $|\tilde{\rho}\rangle$ for the critical density threshold $\tilde{\rho}$. d_c is replaced by d_L and the signs of $\left|d^2_{i,j}\right\rangle - |ans\rangle$ and $\rho_i - \tilde{\rho}$ are additionally

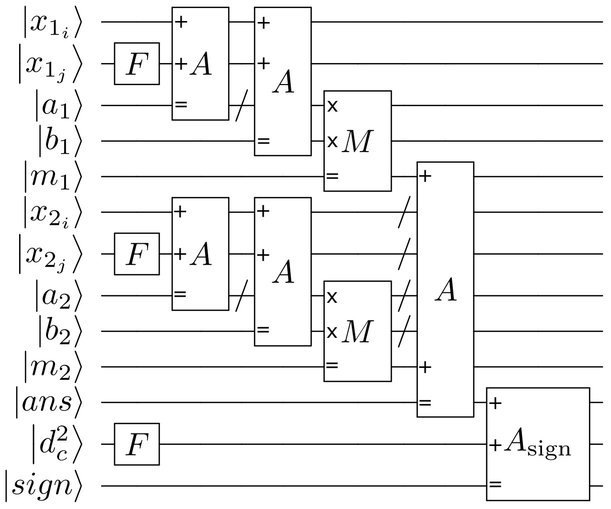

Figure 6: Circuit corresponding to the marking operator U_P in Eq. (11b). First, it computes the Euclidean distance $d_{1,2}$ between points 1 and 2, as defined in Sec. 2.1. Then, it compares $d_{1,2}$ to the threshold d_c and marks the index based on the sign qubit $|sign\rangle$. A and B are defined in Eq. (12) and in Ref. [68]. Bars before gates are employed to indicate that the corresponding qubits are unaffected. In the figure, the first subscript on theregister contains the spatial coordinate and the second subscript contains the index. So, for example, $|x_{1_i}\rangle$ contains the x_1 coordinate of the i^{th} point. The flip gates F flip the sign bit of the input. The ancillas a_1 and b_1 contain the output of computation $x_{1_i} - x_{1_j}$, while a_2 and b_2 contain the output of computation $x_{2_i} - x_{2_j}$, after the first two levels of A gates. The M gates then set m_1 and m_2 to $(x_{1_i} - x_{1_j})^2$ and $(x_{2_i} - x_{2_j})^2$ respectively. The pre-final A gate sets the $|ans\rangle$ state to $(x_{1_i} - x_{1_j})^2 + (x_{2_i} - x_{2_j})^2$. The final A_{sign} gate computes $(x_{1_i} - x_{1_j})^2 + (x_{2_i} - x_{2_j})^2 - d_c^2$. However, for this final A_{sign} gate, we need only compute the sign qubit of the computation and so only the sign qubit computation subcircuit of A features in A_{sign}.

computed in order to find only the points that lie between $d_{i,j}$ and d_{L} and satisfy the $\tilde{\rho}$ threshold. These signs can be combined with the Toffoli gate to compute the required function $f(x)$

For Seeds and Outliers, the condition is similar to that in the Nearest Higher (GEBS) procedure in that the distance criterion is only a lower bound as opposed to a window. The GEBS blackbox without the upper distance bound computation can be used.

For the Cluster Assignment step, the operator U_P needs to check if the Nearest Higher index of the points over which Grover Search is carried out lies within $\mathcal{C}$. The U_P operator for this can be generated using X and C^nZ gates. C^nZ gates map $|1\rangle^{\otimes n}$ to $-|1\rangle^{\otimes n}$ and act as the identity on all the other elements of the computational basis. Let us consider a simple example where the dynamic search space DSS has 8 points indexed from 000 to 111 in binary. So, the Nearest Higher index will always be in this range. If $\mathcal{C}$ consists of indices 0(000) and 5(101), we would use U_P as defined in Fig. 7. This flips the phase of 000 and 101 in the input superposition as required (following Eq. (12b)). As shown in Fig. 7, operators for each index can be sequentially applied to check for multiple indices in $\mathcal{C}$.

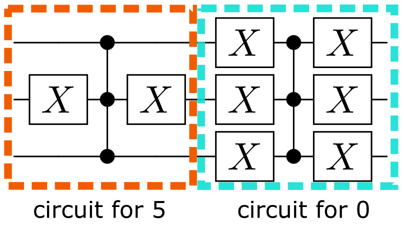

Figure 7: Circuit for U_P to check if the Nearest Higher indices are in $\mathcal{C}$, which here has 2 elements: $5(101)$ and $0(000)$. In the above circuit, the first part of the circuit flips the phase of the state if the register is in the $|101\rangle$ state that corresponds to index 5. The second part flips the phase of the state if the register is in the $|000\rangle$ state that corresponds to index 0. Note that the X gates performed at the end of the two subcircuits are uncomputation steps that ensure that the input to the subsequent subcircuits are as required and unaffected by our initial X gates.

References

[1] Dhruv Gopalakrishnan et al. *QLUE-algo/qlue: frontiers-paper.* Version frontiers-paper. July 2024. DOI: 10.5281/zenodo.12655189.

[2] Zuguang Gu and Daniel Hübschmann. "SimplifyEnrichment: A Bioconductor Package for Clustering and Visualizing Functional Enrichment Results". In: *Genomics, Proteomics and Bioinformatics* 21.1 (June 2022), pp. 190–202. DOI: 10.1016/j.gpb.2022.04.008.

[3] Jelili Oyelade et al. "Data Clustering: Algorithms and Its Applications". In: *2019 19th International Conference on Computational Science and Its Applications (ICCSA).* 2019, pp. 71–81. DOI: 10.1109/ICCSA.2019.000-1.

[4] Tong Wu, Xinwang Liu, Jindong Qin, and Francisco Herrera. "Balance Dynamic Clustering Analysis and Consensus Reaching Process With Consensus Evolution Networks in Large-Scale Group Decision Making". In: *IEEE Transactions on Fuzzy Systems* 29.2 (2021), pp. 357–371. DOI: 10.1109/TFUZZ.2019.2953602.

[5] Giulia Caruso, Stefano Antonio Gattone, Francesca Fortuna, and Tonio Di Battista. "Cluster Analysis as a Decision-Making Tool: A Methodological Review". In: *Decision Economics: In the Tradition of Herbert A. Simon's Heritage.* Ed. by Edgardo Bucciarelli, Shu-Heng Chen, and Juan M. Corchado. Cham: Springer International Publishing, 2018, pp. 48–55.

[6] Girish Punj and David W. Stewart. "Cluster Analysis in Marketing Research: Review and Suggestions for Application". In: *Journal of Marketing Research* 20.2 (1983), pp. 134–148. DOI: 10.1177/002224378302000204.

[7] Jih-Jeng Huang, Gwo-Hshiung Tzeng, and Chorng-Shyong Ong. "Marketing segmentation using support vector clustering". In: *Expert Systems with Applications* 32.2 (2007), pp. 313–317. DOI: https://doi.org/10.1016/j.eswa.2005.11.028.

[8] Xiaohui Wu et al. "Probabilistic Latent Semantic User Segmentation for Behavioral Targeted Advertising". In: *Proceedings of the Third International Workshop on Data Mining and Audience Intelligence for Advertising.* ADKDD '09. Paris, France: Association for Computing Machinery, 2009, 10–17. DOI: 10.1145/1592748.1592751.

[9] Pratik Dutta, Sriparna Saha, Sanket Pai, and Aviral Kumar. "A Protein Interaction Information-based Generative Model for Enhancing Gene Clustering". In: *Scientific Reports* 10.1 (2020), p. 665. DOI: 10.1038/s41598-020-57437-5.

[10] Wai-Ho Au, K.C.C. Chan, A.K.C. Wong, and Yang Wang. "Attribute clustering for grouping, selection, and classification of gene expression data". In: *IEEE/ACM Transactions on Computational Biology and Bioinformatics* 2.2 (2005), pp. 83–101. DOI: 10.1109/TCBB.2005.17.

[11] Jianxin Wang, Min Li, Youping Deng, and Yi Pan. "Recent advances in clustering methods for protein interaction networks". In: *BMC Genomics* 11.3 (2010), S10. DOI: 10.1186/1471-2164-11-S3-S10.

[12] Sitaram Asur, Duygu Ucar, and Srinivasan Parthasarathy. "An ensemble framework for clustering protein–protein interaction networks". In: *Bioinformatics* 23.13 (July 2007), pp. i29–i40. DOI: 10.1093/bioinformatics/btm212.

[13] G.B. Coleman and H.C. Andrews. "Image segmentation by clustering". In: *Proceedings of the IEEE* 67.5 (1979), pp. 773–785. DOI: 10.1109/PROC.1979.11327.

[14] Kishore Kumar R, Lokendra Birla, and Sreenivasa Rao K. "A robust unsupervised pattern discovery and clustering of speech signals". In: *Pattern Recognition Letters* 116 (2018), pp. 254–261. DOI: https://doi.org/10.1016/j.patrec.2018.10.035.

[15] Jianlong Chang, Lingfeng Wang, Gaofeng Meng, Shiming Xiang, and Chunhong Pan. "Deep Adaptive Image Clustering". In: *Proceedings of the IEEE International Conference on Computer Vision (ICCV).* 2017.

[16] Andriy Shepitsen, Jonathan Gemmell, Bamshad Mobasher, and Robin Burke. "Personalized Recommendation in Social Tagging Systems Using Hierarchical Clustering". In: *Proceedings of the 2008 ACM Conference on Recommender Systems*. RecSys '08. Lausanne, Switzerland: Association for Computing Machinery, 2008, 259–266. DOI: 10.1145/1454008.1454048.

[17] Vincent Schickel-Zuber and Boi Faltings. "Using Hierarchical Clustering for Learning Theontologies Used in Recommendation Systems". In: *Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. KDD '07. San Jose, California, USA: Association for Computing Machinery, 2007, 599–608. DOI: 10.1145/1281192.1281257.

[18] "The Phase-2 Upgrade of the CMS Endcap Calorimeter". In: (2017). DOI: 10.17181/CERN.IV8M.1JY2.

[19] G. Aad et al. "Observation of a new particle in the search for the Standard Model Higgs boson with the ATLAS detector at the LHC". In: *Physics Letters B* 716.1 (2012), pp. 1–29. DOI: 10.1016/j.physletb.2012.08.020.

[20] F D Amaro et al. "Directional iDBSCAN to detect cosmic-ray tracks for the CYGNO experiment". In: *Measurement Science and Technology* 34.12 (2023), p. 125024. DOI: 10.1088/1361-6501/acf402.

[21] S A Rodenko, A G Mayorov, V V Malakhov, I K Troitskaya, and on behalf ofPAMELA collaboration. "Track reconstruction of antiprotons and antideuterons in the coordinate-sensitive calorimeter of PAMELA spectrometer using the Hough transform". In: *Journal of Physics: Conference Series* 1189.1 (2019), p. 012009. DOI: 10.1088/1742-6596/1189/1/012009.

[22] Christoph Dalitz et al. "Automatic trajectory recognition in Active Target Time Projection Chambers data by means of hierarchical clustering". In: *Computer Physics Communications* 235 (2019), pp. 159–168. DOI: https://doi.org/10.1016/j.cpc.2018.09.010.

[23] Marco Rovere, Ziheng Chen, Antonio Di Pilato, Felice Pantaleo, and Chris Seez. "CLUE: A Fast Parallel Clustering Algorithm for High Granularity Calorimeters in High-Energy Physics". In: *Frontiers in Big Data* 3 (2020). DOI: 10.3389/fdata.2020.591315.

[24] A. Hayrapetyan et al. "Observation of four top quark production in proton-proton collisions at s=13TeV". In: *Physics Letters B* 847 (2023), p. 138290. DOI: https://doi.org/10.1016/j.physletb.2023.138290.

[25] A. Tumasyan et al. "Measurement of the $B_s^0 \to \mu^+\mu^-$ decay properties and search for the $B^0 \to \mu^+\mu^-$ decay in proton-proton collisions at $\sqrt{s}$=13TeV". In: *Physics Letters B* 842 (2023), p. 137955. DOI: https://doi.org/10.1016/j.physletb.2023.137955.

[26] Aram Hayrapetyan et al. *Search for new physics with emerging jets in proton-proton collisions at $\sqrt{s}$ = 13 TeV*. Tech. rep. Submitted to the Journal of High Energy Physics. All figures and tables can be found at http://cms-results.web.cern.ch/cms-results/public-results/publications/EXO-22-015 (CMS Public Pages). Geneva: CERN, 2024.

[27] R. Aaij et al. "First Observation of a Doubly Charged Tetraquark and Its Neutral Partner". In: *Phys. Rev. Lett.* 131 (4 2023), p. 041902. DOI: 10.1103/PhysRevLett.131.041902.

[28] Lov K. Grover. "A Fast Quantum Mechanical Algorithm for Database Search". In: *Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing*. STOC '96. Philadelphia, Pennsylvania, USA: Association for Computing Machinery, 1996, 212–219. DOI: 10.1145/237814.237866.

[29] Erica Brondolin. *CLUE a clustering algorithm for current and future experiments*. Tech. rep. Geneva: CERN, 2022.

[30] Md Rezaul Karim et al. "Deep learning-based clustering approaches for bioinformatics". In: *Briefings in Bioinformatics* 22.1 (Feb. 2020), pp. 393–415. DOI: 10.1093/bib/bbz170.

[31] Jian Zhou, Linli Zhai, and Athanasios A. Pantelous. “Market segmentation using high-dimensional sparse consumers data”. In: *Expert Systems with Applications* 145 (2020), p. 113136. DOI: https://doi.org/10.1016/j.eswa.2019.113136.

[32] Annie Y. Wei, Preksha Naik, Aram W. Harrow, and Jesse Thaler. “Quantum algorithms for jet clustering”. In: *Phys. Rev. D* 101 (9 2020), p. 094015. DOI: 10.1103/PhysRevD.101.094015.

[33] Iordanis Kerenidis and Jonas Landman. “Quantum spectral clustering”. In: *Phys. Rev. A* 103 (4 2021), p. 042415. DOI: 10.1103/PhysRevA.103.042415.

[34] Iordanis Kerenidis, Jonas Landman, Alessandro Luongo, and Anupam Prakash. “q-means: A quantum algorithm for unsupervised machine learning”. In: *Advances in Neural Information Processing Systems*. Ed. by H. Wallach et al. Vol. 32. Curran Associates, Inc., 2019.

[35] Esma Aïmeur, Gilles Brassard, and Sébastien Gambs. “Quantum Clustering Algorithms”. In: *Proceedings of the 24th International Conference on Machine Learning*. ICML ’07. Corvalis, Oregon, USA: Association for Computing Machinery, 2007, 1–8. DOI: 10.1145/1273496.1273497.

[36] Duarte Magano, Lorenzo Buffoni, and Yasser Omar. “Quantum density peak clustering”. In: *Quantum Machine Intelligence* 5.1 (2023), p. 9. DOI: 10.1007/s42484-022-00090-0.

[37] Davide Nicotra et al. *A quantum algorithm for track reconstruction in the LHCb vertex detector*. 2023.

[38] Cenk Tüysüz et al. “Particle Track Reconstruction with Quantum Algorithms”. In: *EPJ Web of Conferences* 245 (2020). Ed. by C. Doglioni et al., p. 09013. DOI: 10.1051/epjconf/202024509013.

[39] Diogo Pires, Pedrame Bargassa, João Seixas, and Yasser Omar. *A Digital Quantum Algorithm for Jet Clustering in High-Energy Physics*. 2021.

[40] CMS Collaboration. *Review of top quark mass measurements in CMS*. 2024.

[41] CMS Collaboration. *Development of the CMS detector for the CERN LHC Run 3*. 2023.

[42] Bahjat F Qaqish, Jonathon J O’Brien, Jonathan C Hibbard, and Katie J Clowers. “Accelerating high-dimensional clustering with lossless data reduction”. In: *Bioinformatics* 33.18 (May 2017), pp. 2867–2872. DOI: 10.1093/bioinformatics/btx328.

[43] H.P. Ng, S.H. Ong, K.W.C. Foong, P.S. Goh, and W.L. Nowinski. “Medical Image Segmentation Using K-Means Clustering and Improved Watershed Algorithm”. In: *2006 IEEE Southwest Symposium on Image Analysis and Interpretation*. 2006, pp. 61–65. DOI: 10.1109/SSIAI.2006.1633722.

[44] L. Galluccio, O. Michel, P. Bendjoya, and E. Slezak. “Unsupervised Clustering on Astrophysics Data: Asteroids Reflectance Spectra Surveys and Hyperspectral Images”. In: *Classification and Discovery in Large Astronomical Surveys*. Ed. by Coryn A. L. Bailer-Jones. Vol. 1082. American Institute of Physics Conference Series. Dec. 2008, pp. 165–171. DOI: 10.1063/1.3059034.

[45] Michael J. Gaffey. “Space weathering and the interpretation of asteroid reflectance spectra”. In: *Icarus* 209.2 (2010), pp. 564–574. DOI: https://doi.org/10.1016/j.icarus.2010.05.006.

[46] Angela F. Gao et al. “Generalized Unsupervised Clustering of Hyperspectral Images of Geological Targets in the Near Infrared”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*. 2021, pp. 4294–4303.

[47] Jelili Oyelade et al. “Clustering Algorithms: Their Application to Gene Expression Data”. In: *Bioinformatics and Biology Insights* 10 (2016). PMID: 27932867, BBI.S38316. DOI: 10.4137/BBI.S38316.

[48] Vittorio Giovannetti, Seth Lloyd, and Lorenzo Maccone. “Quantum Random Access Memory”. In: *Physical Review Letters* 100.16 (2008). DOI: 10.1103/physrevlett.100.160501.

[49] Gilles Brassard, Peter Høyer, Michele Mosca, and Alain Tapp. *Quantum amplitude amplification and estimation*. 2002. DOI: 10.1090/conm/305/05215.

[50] Fabian Pedregosa et al. *Scikit-learn: Machine Learning in Python*. 2018.

[51] Jekaterina Jaroslavceva. "A New Trackster Linking Algorithm Based on Graph Neural Networks for the CMS Experiment at the Large Hadron Collider at CERN". Presented 14 Jul 2023. Prague, Tech. U., 2023.

[52] Andrew Rosenberg and Julia Hirschberg. "V-measure: A conditional entropy-based external cluster evaluation measure". In: *Proceedings of the 2007 joint conference on empirical methods in natural language processing and computational natural language learning (EMNLP-CoNLL)*. 2007, pp. 410–420.

[53] Michael A. Nielsen and Isaac L. Chuang. *Quantum Computation and Quantum Information: 10th Anniversary Edition*. Cambridge University Press, 2010.

[54] Amine Zeguendry, Zahi Jarir, and Mohamed Quafafou. "Quantum Machine Learning: A Review and Case Studies". en. In: *Entropy (Basel)* 25.2 (Feb. 2023).

[55] Tobias Haug, Chris N Self, and M S Kim. "Quantum machine learning of large datasets using randomized measurements". In: *Machine Learning: Science and Technology* 4.1 (2023), p. 015005. DOI: 10.1088/2632-2153/acb0b4.

[56] Ilya Sinayskiy Maria Schuld and Francesco Petruccione. "An introduction to quantum machine learning". In: *Contemporary Physics* 56.2 (2015), pp. 172–185. DOI: 10.1080/00107514.2014.964942.

[57] Prayag Tiwari, Shahram Dehdashti, Abdul Karim Obeid, Massimo Melucci, and Peter Bruza. *Kernel Method based on Non-Linear Coherent State*. 2020.

[58] Kazuhisa Fujita. "Approximate spectral clustering using both reference vectors and topology of the network generated by growing neural gas". en. In: *PeerJ Comput Sci* 7 (Aug. 2021), e679.

[59] Alessio Celi et al. "Emerging Two-Dimensional Gauge Theories in Rydberg Configurable Arrays". In: *Phys. Rev. X* 10 (2 2020), p. 021057. DOI: 10.1103/PhysRevX.10.021057.

[60] Hannes Bernien et al. "Probing many-body dynamics on a 51-atom quantum simulator". In: *Nature* 551.7682 (2017), pp. 579–584. DOI: 10.1038/nature24622.

[61] Henning Labuhn et al. "Tunable two-dimensional arrays of single Rydberg atoms for realizing quantum Ising models". In: *Nature* 534.7609 (2016), pp. 667–670. DOI: 10.1038/nature18274.

[62] Frank Arute et al. "Quantum supremacy using a programmable superconducting processor". In: *Nature* 574.7779 (2019), pp. 505–510. DOI: 10.1038/s41586-019-1666-5.

[63] B. P. Lanyon et al. "Universal Digital Quantum Simulation with Trapped Ions". In: *Science* 334.6052 (2011), pp. 57–61. DOI: 10.1126/science.1208001.

[64] S. Debnath et al. "Demonstration of a small programmable quantum computer with atomic qubits". In: *Nature* 536.7614 (2016), pp. 63–66. DOI: 10.1038/nature18648.

[65] A. D. Córcoles et al. "Demonstration of a quantum error detection code using a square lattice of four superconducting qubits". In: *Nature Communications* 6.1 (2015), p. 6979. DOI: 10.1038/ncomms7979.

[66] Beatrice Nash, Vlad Gheorghiu, and Michele Mosca. "Quantum circuit optimizations for NISQ architectures". In: *Quantum Science and Technology* 5.2 (2020), p. 025010. DOI: 10.1088/2058-9565/ab79b1.

[67] Jia Wu et al. "Hyperparameter Optimization for Machine Learning Models Based on Bayesian Optimizationb". In: *Journal of Electronic Science and Technology* 17.1 (2019), pp. 26–40. DOI: https://doi.org/10.11989/JEST.1674-862X.80904120.

[68] Raphael Seidel, Nikolay Tcholtchev, Sebastian Bock, Colin Kai-Uwe Becker, and Manfred Hauswirth. *Efficient Floating Point Arithmetic for Quantum Computers.* 2021.