Practical Quantum K-Means Clustering: Performance Analysis and Applications in Energy Grid Classification

STEPHEN DIADAMO[1,2] , **COREY O'MEARA**[1] , **GIORGIO CORTIANA**[1], **AND JUAN BERNABÉ-MORENO**[1]

[1]E.ON Digital Technology GmbH, 45131 Essen, Germany
[2]Technische Universität München, 80333 Munich, Germany

Corresponding authors: Stephen DiAdamo; Corey O'Meara (e-mail: stephen.diadamo@gmail.com; corey.omeara@eon.com)

ABSTRACT In this work, we aim to solve a practical use-case of unsupervised clustering that has applications in predictive maintenance in the energy operations sector using quantum computers. Using only cloud access to quantum computers, we complete thorough performance analysis of what some current quantum computing systems are capable of for practical applications involving nontrivial mid-to-high-dimensional datasets. We first benchmark how well distance estimation can be performed using two different metrics based on the swap-test, using angle and amplitude data embedding. Next, for the clustering performance analysis, we generate sets of synthetic data with varying cluster variance and compare simulation to physical hardware results using the two metrics. From the results of this performance analysis, we propose a general, competitive, and parallelized version of quantum k-means clustering to avoid some pitfalls discovered due to noisy hardware and apply the approach to a real energy grid clustering scenario. Using real-world German electricity grid data, we show that the new approach improves the balanced accuracy of the standard quantum k-means clustering by 67.8% with respect to the labeling of the classical algorithm.

INDEX TERMS Cloud quantum computing, quantum clustering, quantum computing, quantum distance estimation.

I. INTRODUCTION

Given the challenging engineering requirements for building and maintaining quantum computers, it is likely that quantum computers will only be accessible through cloud services for the majority of users. Quantum computers, depending on the qubit technology, can require a complex construction and maintenance schedule that makes it impractical for the average user to own [1]. Rather than building quantum computers as a hardware product to sell to consumers, companies, such as Amazon, Microsoft, and IBM, are rather developing cloud-based platforms for online access to their quantum devices. Although these quantum cloud services are currently accessible, the question of how useable they are for practical industrial use-cases arises. For our interest, we ask: Can these quantum computers produce accurate enough distance estimates for clustering? Can we achieve any speedup with them currently? The focus of this work is, therefore, to benchmark

this simple common use-case using the IBM Quantum cloud services [2].

Clustering algorithms can be used on unlabeled data to find relationships between the data's various features. To perform clustering, an algorithm introduced by Lloyd in 1982, called k-means clustering [3], can be used. The k-means algorithm takes as input a collection of unlabeled data points, or feature vectors, and outputs a list of labels, one for each data point. The data points are labeled based on the minimal distance to a particular centroid. During execution, the algorithm improves the centroid locations by running iteratively, updating the location to be the mean of the data points that are determined nearest to them based on a distance metric.

Classically, the usual method for measuring the distance between centroids and data points is to simply compute the Euclidean distance. For feature vectors of N features, computing the Euclidean distance requires $O(N)$ computational

steps. With a quantum approach, using quantum amplitude embedding (see [4] for details), one can encode length-N vectors into $O(\log_2 N)$ qubits, an exponential decrease in resources for embedding, assuming one can load quantum states into a quantum random access memory [5]. With this embedding, one can perform what is known as a swap-test using a quantum computer, as described in [5] and [6], to compute an estimate for the Euclidean distance between two vectors. Because the swap-test requires a number of operations proportional to the number of qubits used for embedding—needed for swapping two multiqubit states—in theory this would result in an exponential speedup in runtime complexity. Moreover, the minimum distance to a centroid for each point can be found using Grover's search [7] for an additional quadratic speedup. Using a simpler data embedding approach such as angle embedding, the theoretical advantage provided by an amplitude embedding is lost as angle embedding requires a number of qubits directly proportional to the data dimension. However, the benefit of using this alternative embedding is that the depth of the state preparation circuit is constant, whereas with amplitude embedding, the state preparation circuit uses exponentially more nonlocal gates as the data dimension grows [8], leading to vastly deeper circuits using current approaches.

This result makes quantum clustering and nearest-neighbor classification appear very attractive use cases for quantum computing since they both use distance estimation and, therefore, can benefit from a theoretical speedup. When put into practice though, there are various challenges to overcome before one can effectively perform distance estimation on a quantum computer. Moreover, a strong assumption of efficient state preparation needs to be made in order to have an exponential speedup using the quantum approach over the classical approach [9]. Nonetheless, with this article, we aim to expand the results related to clustering on real gate-based quantum hardware and, in particular, to explore how this type of algorithm can be executed on IBM's quantum cloud computing service. In this work, we use only the software libraries and services available to us with no direct hardware access, aiming to demonstrate how well one can expect quantum clustering to perform using generally available resources.

We begin in Section II by reviewing how we perform data encoding and how we calculate the distance estimation in our quantum algorithm using the cloud service. In Section III, we benchmark how well distance estimation can be performed, testing for various Euclidean distances and dimensions in simulation and on hardware. Further, we analyze various clustering experiments using synthetic data with two and four dimensions with several datasets. Given the results of the performance analysis in Section III, we then apply the findings to a nontrivial clustering problem, which is relevant in the energy sector in Section IV. In particular, we show that by decomposing high-dimensional vectors into two-dimensional (2-D) subspace projections, we are able to compute the overall distance in a parallel fashion that significantly reduces the

error induced by existing quantum hardware while simultaneously reducing the total number of circuits. Section V concludes this article.

A. RELATED WORK

Performing clustering and nearest-neighbor type algorithms using quantum computers has been studied in various contexts. Improving the encoding strategy to work better with IBM's quantum computers was studied by Khan et al. [10]. In the article, they describe an encoding mechanism for feature vectors and benchmark the approach on IBM's quantum computer. Feature vectors, after PCA is performed, of dimension two are considered and benchmarked with the MNIST dataset using quantum hardware. Using the IonQ quantum hardware, Johri et al. [11] perform data classification using clustering on their trapped ion quantum computer. In their work, they define an optimized method for encoding 8-D classical data into the quantum computer and use PCA to benchmark against MNIST data for ten different labels and perform their quantum algorithm for nearest-neighbor classification. In these works, an explicit benchmarking of distance estimation accuracy is not demonstrated. Moreover, the only experiments tested on the IBM quantum system were of two dimensions. In a quantum annealing setting, clustering has also been considered. Kumar et al. [12] and Arthur et al. [13] map a clustering problem to a quadratic unconstrained binary optimization problem for an adiabatic quantum computer and use hardware to test their approach. Quantum annealing uses a different approach to quantum computing versus the gate-based model, indeed no swap-test is involved, and the results from annealing experiments do not paint a clear picture of performance using a universal quantum computing approach.

In [14], Benlamine et al. review three methods for distance estimation using a quantum approach and benchmark the approaches using nearest-neighbor classification in simulation only, not testing their approaches on real hardware. Further modified quantum clustering approaches were proposed in [15] and [16]. These works did not perform tests on real physical hardware and, therefore, did not benchmark their performance, as we do in this work. In [17], Nguyen et al. run experiments to test the accuracy of the swap-test using their trapped ion system in a continuous variable setting but do not perform any experiments of clustering or distance estimation for classical input vectors. Overall, our work is the first to benchmark these popular swap-test-based distance estimation techniques in a general way and for the purpose of classification and clustering.

B. SUMMARY OF CONTRIBUTIONS

In this work, we analyze how well current quantum computers can perform unsupervised clustering. To do this, we use two different distance metrics that are based on the swap-test and thoroughly benchmark their accuracy. The swap-test is a general method for computing distance between vectors on a quantum computer and is, therefore, important to thoroughly understand. The two metrics we use are based on popular

well-known methods of data embedding and, therefore, of general interest to explore, something this work does for the first time on real quantum hardware. To compare the accuracy of the real quantum device to the true result, we use noiseless simulation as the ground truth. In order to perform many distance measurements at once for k-means clustering, we define two approaches to parallelize the calculations. In the first case, we offer a parallelization approach when the quantum computing platform does not allow for sending circuits to execute in bulk. We tested these approaches to perform benchmarking, but because of the large performance improvement when using bulk circuit execution of the second approach, we use only the second approach for our analysis in this work. We benchmark distance estimation via the swap-test, varying the scale of the distance, the number of shots used, and the dimension of the data. To the best of the authors' knowledge, no such benchmarking has been previously demonstrated.

Next, we analyze unsupervised clustering on real quantum devices. Using the results of the distance metric analysis, we benchmark the best cases in terms of dimensions—2-D and 4-D—using synthetic data. We found that the accuracy of the real devices using standard approaches and synthetic data proved relatively low. We next tested the ability to perform a true unsupervised clustering problem using real data from the German energy grid. We found that using a standard approach for quantum k-means clustering, the accuracy result output from real quantum hardware is low. We then tested our novel approach to perform the distance metrics, which is to decompose the distance calculation into smaller dimensional projections, and then use quantum circuits that compute multiple swap-tests at once, thereby accommodating the additional swap-tests required after projecting to smaller dimensions. With this approach, we were able to achieve high accuracy with respect to the classical results, thus demonstrating a scalable approach to distance estimation with high accuracy on real quantum devices.

II. PROGRAM SETUP AND CONFIGURATION

In this section, we review the setup and configuration used to perform clustering algorithms. In Section II-A, we review how the synthetic data are generated. Next, we review how the quantum algorithm works as well as the two methods used for classical data embedding in quantum states in Section II-B. Finally, we review the software approach used to execute the algorithms on the quantum cloud hardware in Section II-C.

A. GENERATING SYNTHETIC DATA AND QUANTUM DATA LOADING

In the experiments conducted, we used synthetic data generated with varying dimensions, number of clusters, cluster variance, and minimal distance between the centers. At a high level, the cluster generation algorithm used works by first selecting k center points to then generate cluster data around them. Using the center points as the multidimensional
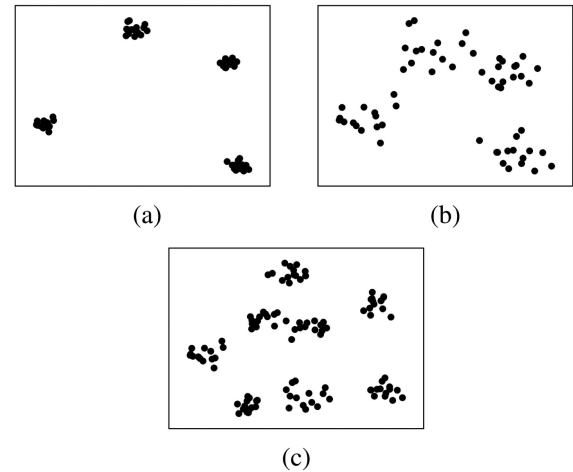

FIGURE 1. Synthetic 2-D data used for clustering. (a) Data are tightly clustered with four clusters with 60 points in total, which we consider an *easy* clustering. (b) Data are more scattered with four clusters and 60 points in total, which we consider a *hard* clustering. (c) Eight clusters with 14 points per cluster for a total of 112 points.
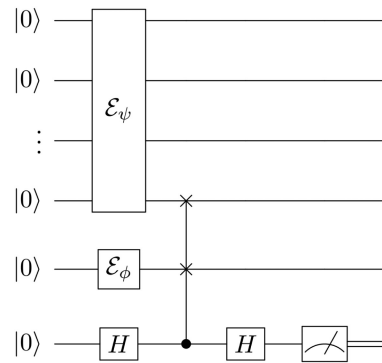

FIGURE 2. Circuit for embedding classical data using amplitude embedding. We make use of the built-in features of Qiskit to initialize the amplitude encoded data.

mean of a multivariate normal distribution, a set number of cluster points are generated surrounding the center. Input to this algorithm is a cluster-variance parameter that we use to set the variance level of the respective dimension to control the "tightness"—how we measure the difficulty of clustering—of the cluster. To avoid the randomly initialized center points being too close to each other, an additional step that resets a center point if it is within some ϵ distance from the already initialized center points is added. The synthetic data generated can be seen in Fig. 1.

To perform the quantum distance estimation algorithm, the generated data points first need to be embedded into a multiqubit quantum state. For this, we use two common types of embeddings, namely amplitude embedding and angle embedding [4], and test them independently. To perform the embeddings, we use the circuit structure shown in Figs. 2 and 3, respectively. To implement amplitude embedding in code, we use the built-in `initialize` function offered by Qiskit, which is a function that takes as input a real vector and returns the necessary gate set for complex amplitude
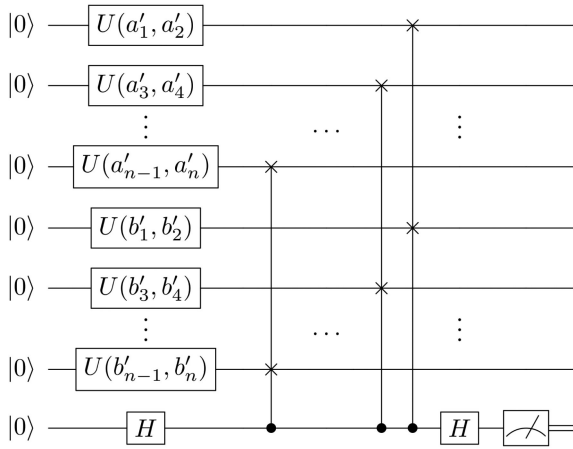
FIGURE 3. Circuit for embedding classical data using angle embedding with swap test.

embedding [18]. For angle embedding, we use 2-D rotations to embed two dimensions of a data vector per one qubit. This type of embedding is also referred to as "dense angle embedding" [19], but in this work, we will only use the name "angle embedding" to refer to it. The scaling in terms of circuit depth and number of two-qubit gates varies significantly between the two embeddings. When embedding data using angle embedding, gate depth does not increase with the data dimension but circuit width grows linearly. On the other hand, embedding classical data using amplitude embedding can scale quite poorly in depth with respect to the data dimension [8]. When using NISQ-era quantum devices, width and depth are some of the properties that should be reduced as far as possible to reduce the effects of noise in any algorithm. We plot the experimental scaling with respect to the data dimension in Fig. 4. The figure shows both the circuit depth and number of nonlocal gates required for initializing data for a connected quantum computer (i.e., one in which any two qubits can interact without any state-swapping) (solid line), and the 65-qubit IBM Brooklyn topology (dashed line), followed by a swap-test. For the embedding methods, Fig. 5 shows the qubit resource requirement trends against the feature vector dimension. The connectivity of the quantum computer topology determines the number of local swaps needed to perform two qubit gates, which results in deeper circuits with more nonlocal gates.

B. QUANTUM CIRCUITS FOR DISTANCE ESTIMATION

To perform clustering, we replace the distance calculation from the classical algorithm with a quantum algorithm for distance estimation. Using amplitude embedding, we use a Euclidean distance approximation based on the one developed in [5] and reiterated in [20]. For angle embedding, we define a simple encoding that scales the data for embedding. The algorithm used for distance estimation first embeds the data for two vectors and then performs a swap-test. To approximate the distance, a number of repetitions, or shots, of the circuit are used to aggregate measurement statistics for
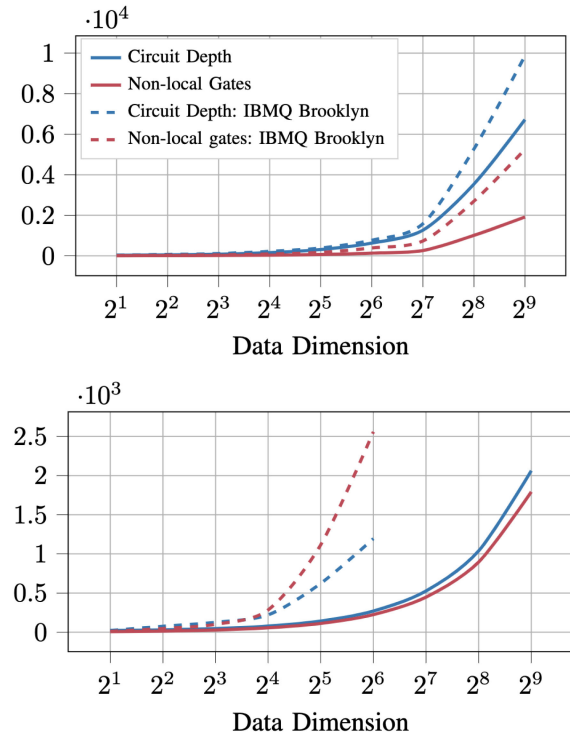

FIGURE 4. Circuit-depth and number of nonlocal gates in a circuit that embeds classical data using amplitude embedding and performs the swap test of varying dimension transpiled with the basis gates under full connectivity and for the IBMQ Brooklyn device. The upper plot demonstrates results using amplitude embedding, and the bottom plot from angle embedding.
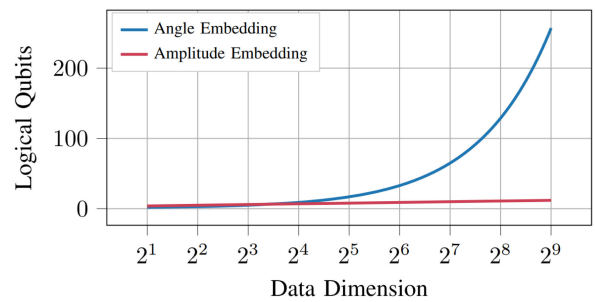

FIGURE 5. Number of qubits required to embed classical data of varying dimension.

a single ancilla qubit. The number of repetitions to use will vary from problem to problem and hardware to hardware. Theoretically, using a higher number of shots will produce a more precise distance estimates, but each hardware will have a finite precision, and so more shots do not always lead to more accuracy. Moreover, some datasets do not require high-precision distance estimation to cluster accurately if they, for example, have well-separated clusters. We explore this further. Once the repetitions are complete, the distance estimation can be calculated via the probabilities of a 0 or 1 measurement outcome of the ancilla qubit

$$\Pr(0) = \frac{1}{2} + \frac{1}{2} |\langle \psi | \phi \rangle|^2 \qquad (1)$$

$$\Pr(1) = 1 - \Pr(0) = \frac{1}{2} - \frac{1}{2} |\langle \psi | \phi \rangle|^2 . \qquad (2)$$

Thus, the estimate $|\langle \psi | \phi \rangle|^2$ provides an approximation to the inner product for our choice of embedding or at least a measure that scales with the inner product in the case of angle embedding.

For each of the two embeddings used, the swap-test procedure differs in terms of the number of controlled swaps—also known as Fredkin gates [21]—used due to the difference in qubit resources required to perform the data embedding as well as the data representation strategy. For two (not necessarily normalized) data vectors $a := (a_1, a_2, \ldots, a_n)$ and $b := (b_1, b_2, \ldots, b_n)$—which in the case for clustering or classification would be one centroid and one data point—the two quantum states that are compared when performing the swap-test when using amplitude embedding are given by

$$|\psi\rangle := \frac{1}{\sqrt{2}} (|0\rangle |a\rangle + |1\rangle |b\rangle) \qquad (3)$$

$$|\phi\rangle := \frac{1}{\sqrt{Z}} (|a||0\rangle - |b||1\rangle) \qquad (4)$$

where $Z := |a|^2 + |b|^2$ [5]. To recover the distance estimation in this case, an ancilla qubit initialized in the $|0\rangle$ state is added to the system, and, after applying a Hadamard gate to it, a Fredkin (or controlled-swap) gate is applied with the ancilla as the control with $|\psi\rangle$ and $|\phi\rangle$ as the target systems. In this case, since $|\phi\rangle$ is just one qubit, one Fredkin gate is needed, thereby greatly reducing the number of controlled swaps needed. Another Hadamard gate is applied to the ancilla qubit and then it is measured. By repeating the process a number of times, $\Pr(0)$ can be estimated to finally recover the distance

$$|a - b|^2 = 4Z(\Pr(0) - 0.5) . \qquad (5)$$

The circuit for this process is depicted in Fig. 2.

For the angle embedding version of distance estimation, we prepare the two angle-encoded data vectors in the following way. Given the classical input vectors a and b defined above, let

$$a_i' := \frac{\pi}{2}(a_i + 1) \qquad (6)$$

$$b_i' := \frac{\pi}{2}(b_i + 1) \qquad (7)$$

such that the angle embedded representations are given by $a' := (a_1', a_2', \ldots, a_n')$ and $b' := (b_1', b_2', \ldots, b_n')$. The mappings given by (6) and (7) are used to ensure that the values are between 0 and π. With this embedding, we can encode data using a 2-D rotation operation defined by

$$U(\theta, \gamma) := \begin{pmatrix} \cos \frac{\theta}{2} & -\sin \frac{\theta}{2} \\ e^{i\gamma} \sin \frac{\theta}{2} & e^{i\gamma} \cos \frac{\theta}{2} \end{pmatrix} . \qquad (8)$$

We encode the two vectors a' and b' in $\lceil n/2 \rceil$ qubits, using 1 qubit per two dimensions of each vector resulting in the final

states

$$|\psi\rangle := \bigotimes_{i \in \mathrm{odd}(n)} U(a_i, ' a_{i+1}')|0\rangle \qquad (9)$$

$$|\phi\rangle := \bigotimes_{i \in \mathrm{odd}(n)} U(b_i, ' b_{i+1}')|0\rangle \qquad (10)$$

where $\mathrm{odd}(n)$ is the set of odd numbers from 1 to n. To recover the distance estimation, again an ancilla qubit is introduced to the system. A Hadamard gate is applied to the ancilla followed by a series of $n/2$ controlled-swap gates using the ancilla as the control and one qubit from $|\psi\rangle$ and one from $|\phi\rangle$. A final Hadamard gate is applied to the ancilla qubit and then is measured. The overall circuit is depicted in Fig. 3. The goal in this case is to produce an estimate for $\Pr(1)$, which is a valid distance metric. Because we are using arbitrary data that are not normalized in advance, we accommodate for this in the distance metric. For this, we set $Z := |a|^2 + |b|^2$ such that a and b are normalized as $a = a/\sqrt{Z}$ and $b = b/\sqrt{Z}$. The final distance metric is, therefore, given by

$$d(a, b) = \sqrt{Z \cdot P(1)} \qquad (11)$$

where $P(1)$ is given in (2). We make the choice not to consider prenormalized data in this case because when using amplitude embedding with the swap-test, by design there is no need to prenormalize data. For (dense) angle embedding, a good practice is to consider, based on the dataset, how to divide the interval $[0, \pi)$ into subintervals such that each subinterval has leeway to accommodate overrotation due to imperfect quantum gates when embedding the data. This would affect the definitions for (6) and (7) for each unique dataset we use, as we would need to change the coefficient and the linear shift to accommodate the largest distance between data points in the dataset. Therefore, for this work, we do not normalize the data ahead of time to better align with the amplitude embedding approach and to keep the embedding process general for varying datasets.

C. RUNNING CLUSTERING ON THE QUANTUM CLOUD
To implement the circuit preparation and execution of the quantum circuits in software, we use IBM's Qiskit Python software development kit (SDK) [22].

A property of the k-means clustering algorithm is that it is highly parallelizable, and we use this property to execute the clustering algorithm more time-efficiently. In each iteration of k-means clustering, a distance is calculated between the current set of centroids and each point in the dataset. These distance calculations are independent of each other and can be computed in parallel using multiprocessing or in a batch job. In the quantum case, to parallelize, we take two approaches. First, we prepare one circuit per distance estimation and then send the collection of circuits as a batch job to the cloud service. The response is the measurement results of the circuits which can then be postprocessed according the embedding. The second way we parallelize the algorithm is to embed multiple swap-tests into one circuit and execute

them on the same quantum computer at once. This allows for multiple distance estimations to be done on a single quantum computer.

We developed the first parallel approach in two ways, one using our own approach and later, when it became available, using a Qiskit native approach. For the first approach, we use many computational thread processes (locally) to send requests simultaneously to the server to reduce computation time. In the simulation, the approach improved performance, but when there is only one quantum computer, this makes no difference. The advantage of parallel execution is achieved with multiple quantum computers working together to compute distance estimations as described in [23].

The second approach was sending batches of circuit jobs via a recently added feature called Qiskit Runtime. This feature allows the user to submit many circuit jobs at once such that there is only one job to queue. The performance improvement between parallel execution via multiprocessing and batch-circuit execution that we observed was significant. The complete algorithms for the two applied k-means clustering approaches are described by Algorithms 1 and 2, respectively.

III. BENCHMARKING ON QUANTUM HARDWARE

To develop a clear understanding of how well clustering and nearest-neighbor classification can be performed using current quantum hardware, it is helpful to test how accurately distance estimation can be performed on quantum devices. In this section, we rigorously investigate the performance of distance estimation between vectors of different dimension sizes, number of shots used, standard embedding approaches, and vector distances. Building on this distance measure study, we then benchmark quantum k-means clustering which uses the previously described distance calculation by means of low- to mid-dimensional synthetic data and clusters.

A. BENCHMARKING DISTANCE ESTIMATION

In this section, we benchmark a variety of cases for performing distance estimation and compare the estimations using simulation and real quantum hardware. Simulation, in this case, is done using the noiseless quantum computing simulation platform offered via the Qiskit framework using the OpenQASM backend. Software for the simulation is prepared in exactly the same way in which the quantum circuit logic is sent to the quantum cloud services, we can simply switch the target backend from the simulated backend to the hardware. To perform the analysis, we test three different cases. We first test how varying the number of circuit shots affects the estimation in both two and four dimensions. Next, we vary the distance of the data points also in two and four dimensions. Finally, we test how varying the data dimensions affects the estimation of up to 32 dimensions using amplitude embedding and up to 26 dimensions using angle embedding.

In all tests, for each resulting data point, we use 100 repetitions, plotting the average output and the standard deviation.

Algorithm 1: Clustering on the Quantum Cloud With Multiprocessing.

Input:
- k: The number of clusters
- data: The data to cluster
- embedding: The choice of data embedding
- ϵ: The minimum distance between two cluster centers
- *maxIterations*: The maximum number of iterations to make
- *processes*: A list of running processes

Output: An ordered list of labels for the data points

1:    *centroids* $\leftarrow$ initialize the centroids using ϵ min distance
2:    *convergence* $\leftarrow$ false, $i \leftarrow 0$
3:    **while** not *convergence* **do**
4:      *circuits* $\leftarrow$ generate all circuits with the data, centroids, and embedding choice
5:      *dists* $\leftarrow$ initialize empty shared storage
6:      **while** not all circuits have been processed **do**
7:        **if** a process is idle **then**
8:          *job* $\leftarrow$ send single circuit to server and await response
9:          *dist* $\leftarrow$ process *job* results according to the embedding
10:          add *dist* with circuit number to shared storage
11:        **else**
12:          wait
13:        **end if**
14:      **end while**
15:      Sort *dists* according to circuit number
16:      *dists* $\leftarrow$ using the returned, ordered measurement results from the server, complete the distance estimation procedure
17:      *labels* $\leftarrow$ using the distances to the centroids, label the data points
18:      *centroids* $\leftarrow$ with the updated labels, recompute the centroids as an average position of the labeled data. Delete centroids for empty clusters.
19:      Check for centroid convergence, update *convergence*
20:      $i \leftarrow i + 1$
21:      Check if $i = $ *maxIterations* and break accordingly
22:    **end while**
23:    **return** *labels*

In each case, we use the measurement error mitigation feature when running the experiments on the physical hardware. To execute the 100 instances, we make 100 copies of the circuits and use the circuit-runner service to execute the circuits in a batched job.

Algorithm 2: Clustering on the Quantum Cloud With Batched Circuits.

> **Input:**
> - k: The number of clusters
> - data: The data to cluster
> - embedding: The choice of data embedding
> - ϵ: The minimum distance between two cluster centers
> - *maxIterations*: The maximum number of iterations to make
>
> **Output:** An ordered list of labels for the data points

1:     *centroids* ← initialize the centroids using ϵ min distance
2:     *convergence* ← false, $i \leftarrow 0$
3:     **while** not *convergence* **do**
4:        *circuits* ← generate all circuits with the data, centroids, and embedding choice
5:        *job* ← send the collection of circuits to the cloud server for processing and await response
6:        *dists* ← using the returned, ordered measurement results from the server, complete the distance estimation procedure
7:        *labels* ← using the distances to the centroids, label the data points
8:        *centroids* ← with the updated labels, recompute the centroids as an average position of the labeled data. Delete centroids for empty clusters.
9:        Check for centroid convergence, update *convergence*
10:     $i \leftarrow i + 1$
11:     Check if $i = maxIterations$ and break accordingly
12:     **end while**
13:     return *labels*

The results of varying the number of circuit shots are plotted in Fig. 6. In this experiment, we created a circuit for the vectors (1,0) and (1,1). Comparing simulation to the physical device results in simulation, we see a convergence in the number of shots to the true answer, and moreover, the average is close to the true distance as desired. With the real hardware, we see no convergence trends behind the dotted line, and using more shots than around 2000 does not generally perform better than using the maximum number of shots 8192. These behaviors indicate that the output probability in the measurements using the real hardware is not consistently Gaussian as one would expect from a noiseless system. Thus, convergence between simulation and real hardware does not match their trends. In some cases, for the same reasons, a lower number of shots performed better, having a lower

variance in the standard deviation than with more shots, as seen, for example, in Fig. 6, where the standard deviation is smaller with fewer used shots indicating more consistent outputs overall. Measurement consistency is important for iterative algorithms when many measurements are used, and so a small standard deviation of the estimate is critical. On the other hand, when we switched to a quantum computer that supported a higher number of shots, much better and more consistent results are seen. For 4-D, the points (1,0,0,0) and (1,1,1,1) are used, and the same effects are more or less seen. Convergence is not reached using real hardware, but a relatively rapid convergence is seen in simulation, where with a high number of shots, the results improve significantly. We reiterate that in some cases, depending on the dataset, high-precision distance estimation is not necessary for clustering, and therefore, this level of accuracy could suffice in some cases.

An important point of note is that for angle embedding, the difference in output between the simulation and the hardware is much starker than with amplitude embedding. The reason behind this is due to the fact that on the IBM quantum devices, the accuracy of qubit rotations is less precise with the available gate basis of $[CX, ID, RZ, SX, X]$ and, therefore, significant differences between the simulation, where such rotations are highly precise, are observed [24]. Another noteworthy aspect is that in some instances, the variance in the standard deviation can become very small for the 100 samples, with no recognizable trend. See, for example, Fig. 6, the 7000 shots point in the upper-left plot and the 6000 shot point in the lower-left plot. We suspect this may come from a periodic hardware calibration that is performed by IBM that was executed on the quantum devices between experimental runs.

For the next set of experiments, we vary the distance between two points in 2-D and 4-D (see Fig. 7). The vectors we chose have the form $(1, \ldots, 1)$ and $(x, \ldots, x)$, where we vary x to modify the distance between the points. The results of the 2-D experiments show that the simulation and the physical device have similar outputs for $x \leq 5$, but for $x > 5$, the outputs from the simulation and real devices start to diverge. In 4-D, the effects of embedding show a stark difference between amplitude and angle embeddings. Interestingly, the simulation results for 4-D amplitude embedding match very closely to the hardware execution for all tested values of x, more so even than in 2D. On the other hand, angle embedding performs far worse in 4-D than in 2-D, where already for $x = 5$, the difference between simulation and real hardware is significant. This extra noise is again likely due to the fact that for more dimensions, the limited precision of the angle embedding is now applied for another two dimensions and that an additional controlled-swap is introduced.

In the last set of experiments, we vary the dimension of the data to observe the limits to the number of features we can use and with what level of accuracy. We test up to 32 dimensions for amplitude embedding and up to 26 dimensions for angle
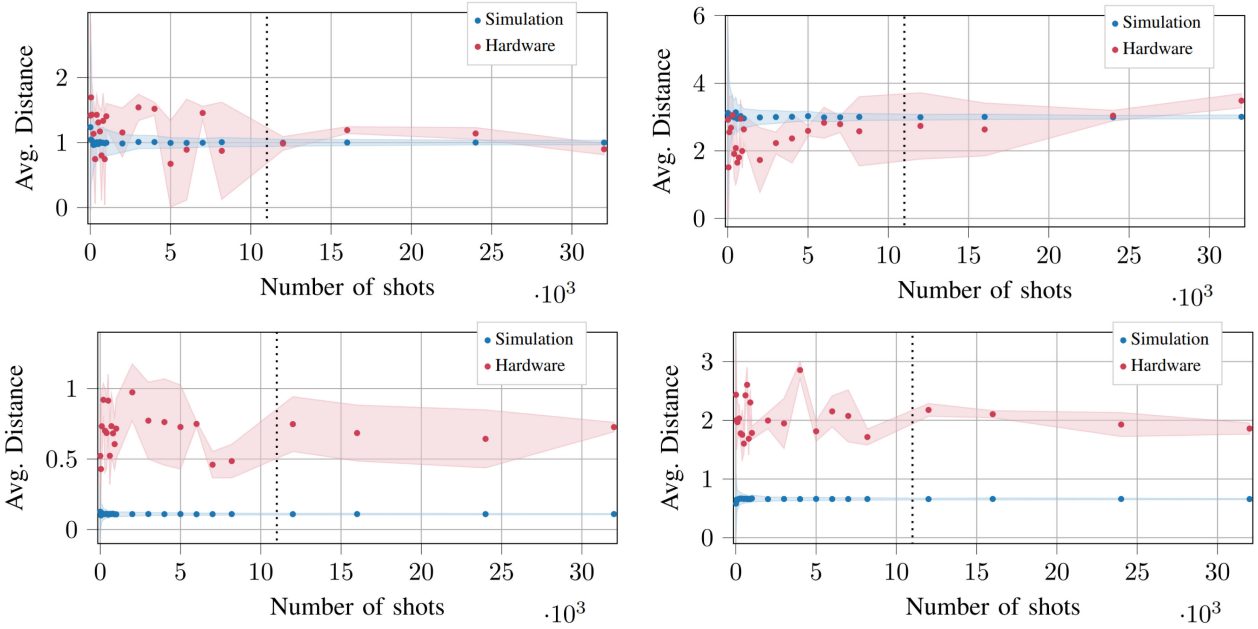
FIGURE 6. Distance estimation for amplitude embedding (upper) and angle embedding (lower) of $(1, 0)$ and $(1, 1)$ in 2-D (left) and $(1,0,0,0)$ and $(1,1,1,1)$ in 4-D (right). The circuit for measuring the distance between these two points is generated and ran a varying number of shots. Displayed is the average output of 100 trials with the standard deviation shaded around the average. The black dotted line indicates where the quantum hardware used switches from IBMQ Bogata on the left of the line to IBMQ Casablanca on the right.
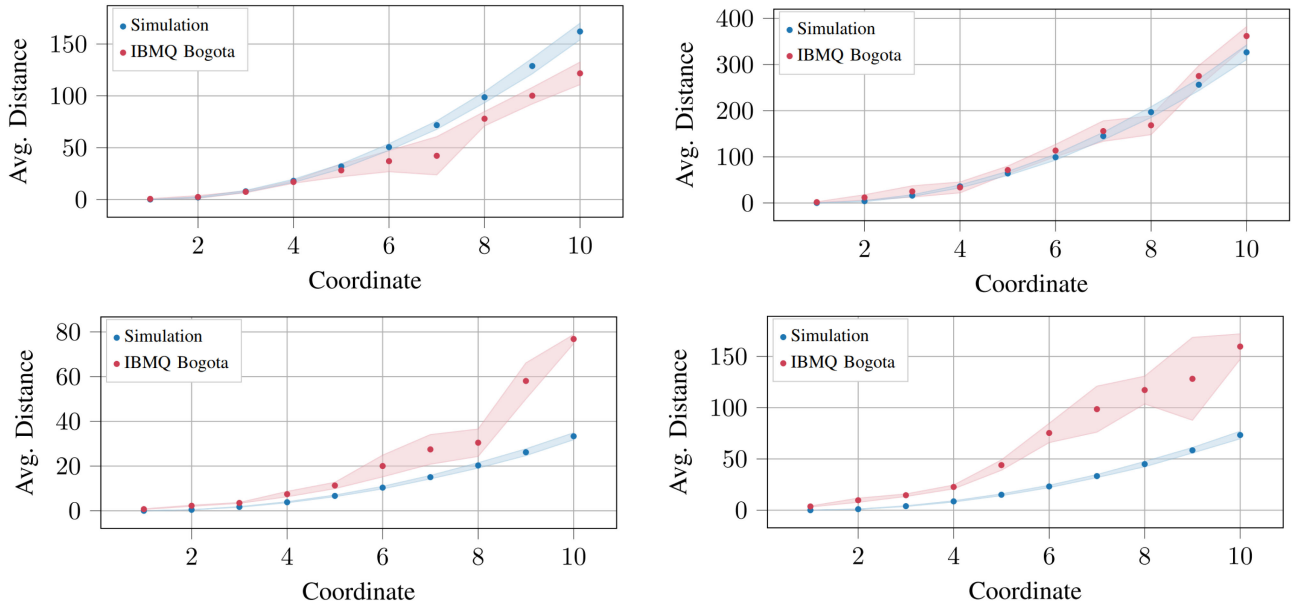

FIGURE 7. Plots for varying the distance between points. The plots on the left are for amplitude (top) and angle (bottom) embedding in two dimensions. The right plots are for amplitude (top) and angle (bottom) embedding in four dimensions. We run the experiments for a vector of shape $(1, \ldots, 1)$ for the base points and, $(x, \ldots, x)$ for the varying point. We repeat the experiments 100 times with 2048 shots plotting the average with the standard deviation.

embedding, using the 27 qubit device IBMQ Sydney to its capacity in the angle embedding case. We compare vectors with shape $(1, \ldots, 1)$, to vectors $(2, \ldots, 2)$, $(3, \ldots, 3)$, and $(4, \ldots, 4)$. In Tables 1 and 2, we show the percent difference between the simulation outputs and the results from the quantum computer for amplitude and angle embedding respectively, whereas Fig. 8 displays the corresponding results. Additionally, we show the average gate depth, single-qubit

gates, and two qubit gates required to execute the circuit. Circuit transpiling and optimization is a randomized process [25] and so we write the standard deviation for these values after transpiling 100 times. The results demonstrate a clear relationship between total gate depth, 2-qubit gate counts, and the data dimensions effect on distance estimation. Although distance estimation using either amplitude or angle embedding approaches each have different gate-depth
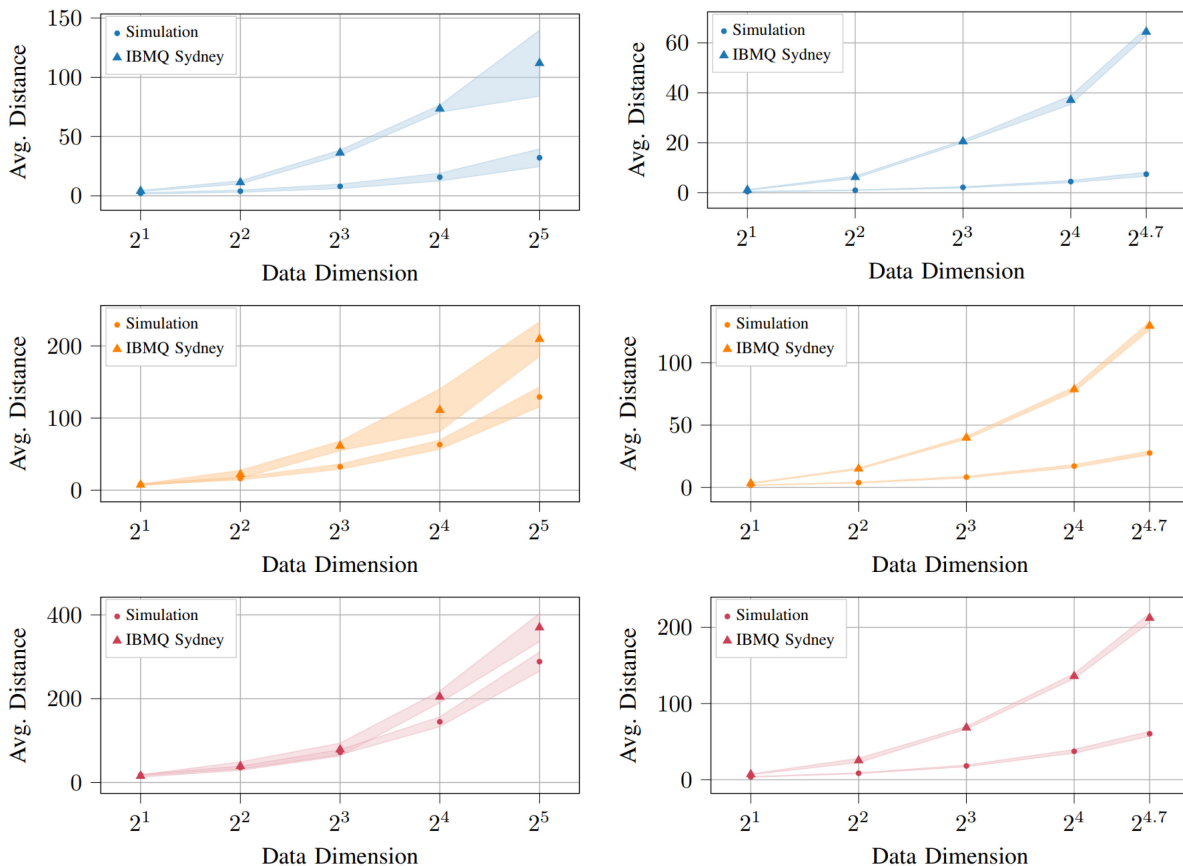
FIGURE 8. Results for varying data dimension using the vector $(1, \ldots, 1)$, estimating the distance to vectors $(2, \ldots, 2)$, $(3, \ldots, 3)$, $(4, \ldots, 4)$. The plots are ordered respectively from top to bottom. We run the experiment 100 times with 2048 circuit shots and plot the average output and standard deviation of simulation after running on the IBMQ Sydney device for up to 32 dimensions with amplitude embedding (left) and 26 dimensions with angle embedding (right).

TABLE 1 Amplitude Embedding: Percent Difference Comparison Between Simulation and Real Data for the Distance Between Data Points of Varying Dimension and Distance on IBMQ Sydney

Dim.	$x = 2$	$x = 3$	$x = 4$	Depth	1Q	2Q
2	66.4%	9.6%	14.0%	25 ± 1	28	12 ± 1
4	97.9%	30.3%	9.2%	37 ± 4	31	25 ± 5
8	128.2%	61.4%	8.4%	50 ± 4	34	39 ± 5
16	133.2%	55.0%	34.1%	93 ± 8	37	84 ± 9
32	110.9%	47.4%	24.7%	167 ± 10	40	173 ± 14

We write the average circuit depth, number of 1-qubit gates and number of 2-qubit gates. Note that circuit optimization is a randomized process, but with a number of fixed single qubit gates [25], and therefore, we show the standard deviation of 100 independent circuit optimizations.

TABLE 2 Angle Embedding: Percent Difference Comparison Between Simulation and Real Data for the Distance Between Data Points of Varying Dimension and Distance on IBMQ Sydney

Dim.	$x = 2$	$x = 3$	$x = 4$	Depth	1Q	2Q
2	85.9%	61.9%	61.3%	25 ± 0	27	11 ± 0
4	144.2%	117.9%	98.8%	53 ± 3	47	42 ± 4
8	161.7%	131.1%	115.9%	89 ± 7	87	99 ± 10
16	156.8%	128.4%	113.8%	209 ± 17	167	364 ± 40
26	158.6%	129.6%	111.4%	328 ± 22	267	611 ± 50

We write the average circuit depth, number of 1-qubit gates and number of 2-qubit gates. Note that circuit optimization is a randomized process, but with a number of fixed single qubit gates [25], and therefore, we show the standard deviation of 100 independent circuit optimizations.

amounts, we observe empirical evidence of the decoherence effects as a function of problem (data) size. We also observe that at further distances, simulation and hardware results tend to agree more closely in both embedding types. We believe that the accuracy tends to improve as the true distances increase because the normalization factor has a bigger impact on the postprocessing, thereby better mitigating inaccurate distance estimates. These results motivate data scaling techniques, where mapping the data into a space where the data are more separated could result in more accuracy, a hypothesis we intend to explore in future work.

B. BENCHMARKING QUANTUM CLUSTERING

In this section, we benchmark the quantum k-means clustering algorithm using various dimensions and number of clusters using synthetic data. To determine the accuracy of clustering using the quantum approaches, we generated three types of synthetic data using two and four dimensions. The first two types of data that we generate are an *easy* dataset, a *hard* dataset both with four clusters and 15 data points in each cluster, as seen in Fig. 1(a) and (b). This totals 60 data points. The third dataset we use has a variance between the *easy* and *hard* sets but with eight clusters and 14 points per
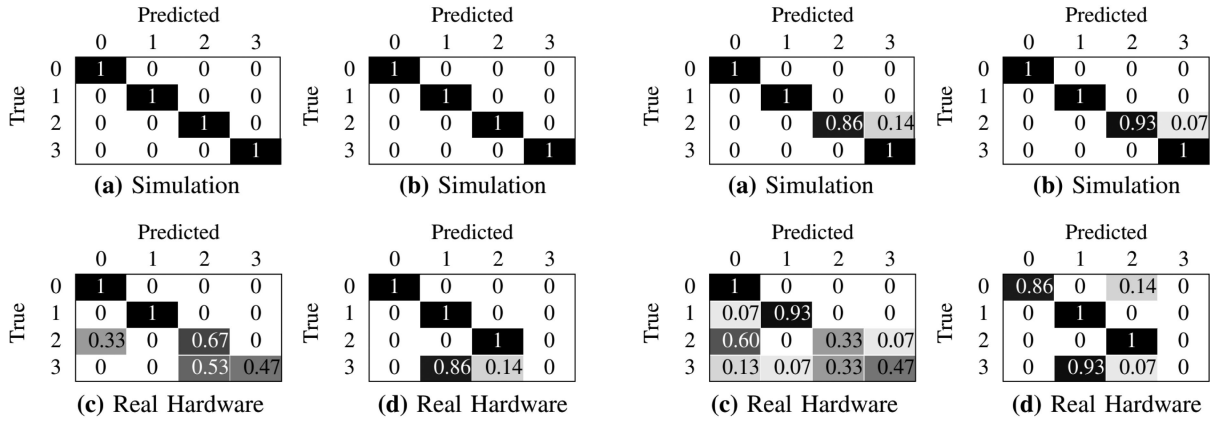
(a) Simulation

True \ Predicted	0	1	2	3
0	1	0	0	0
1	0	1	0	0
2	0	0	1	0
3	0	0	0	1

(b) Simulation

True \ Predicted	0	1	2	3
0	1	0	0	0
1	0	1	0	0
2	0	0	1	0
3	0	0	0	1

(a) Simulation

True \ Predicted	0	1	2	3
0	1	0	0	0
1	0	1	0	0
2	0	0	0.86	0.14
3	0	0	0	1

(b) Simulation

True \ Predicted	0	1	2	3
0	1	0	0	0
1	0	1	0	0
2	0	0	0.93	0.07
3	0	0	0	1

(c) Real Hardware

True \ Predicted	0	1	2	3
0	1	0	0	0
1	0	1	0	0
2	0.33	0	0.67	0
3	0	0	0.53	0.47

(d) Real Hardware

True \ Predicted	0	1	2	3
0	1	0	0	0
1	0	1	0	0
2	0	0	1	0
3	0	0.86	0.14	0

(c) Real Hardware

True \ Predicted	0	1	2	3
0	1	0	0	0
1	0.07	0.93	0	0
2	0.60	0	0.33	0.07
3	0.13	0.07	0.33	0.47

(d) Real Hardware

True \ Predicted	0	1	2	3
0	0.86	0	0.14	0
1	0	1	0	0
2	0	0	1	0
3	0	0.93	0.07	0

FIGURE 9. Confusion matrices for the execution for clustering the *easy* (right) and *hard* datasets (left) with two features using amplitude (left column) and angle (right column) embeddings. (a) and (b) Clustering outcome of the noiseless quantum simulation. (c) and (d) Results of running on the real IBMQ Sydney device. Each instance ran for a maximum of five iterations using the maximum number of shots 8,192.
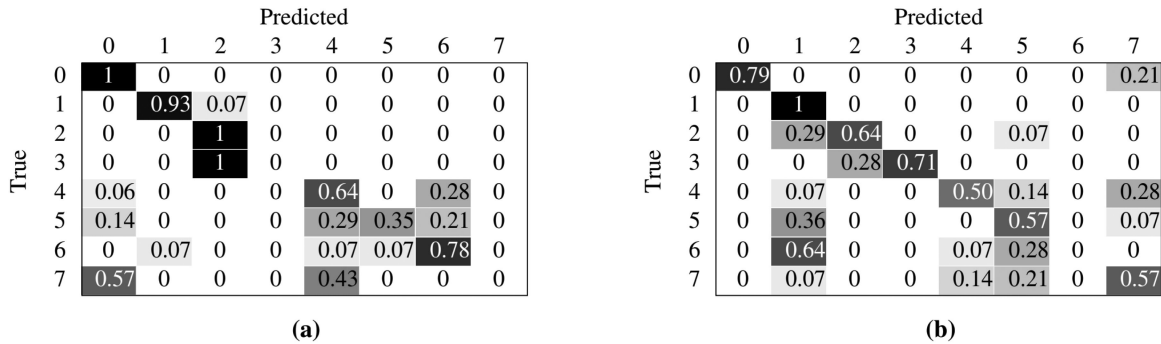
(a)

True \ Predicted	0	1	2	3	4	5	6	7
0	1	0	0	0	0	0	0	0
1	0	0.93	0.07	0	0	0	0	0
2	0	0	1	0	0	0	0	0
3	0	0	1	0	0	0	0	0
4	0.06	0	0	0	0.64	0	0.28	0
5	0.14	0	0	0	0.29	0.35	0.21	0
6	0	0.07	0	0	0.07	0.07	0.78	0
7	0.57	0	0	0	0.43	0	0	0

(b)

True \ Predicted	0	1	2	3	4	5	6	7
0	0.79	0	0	0	0	0	0	0.21
1	0	1	0	0	0	0	0	0
2	0	0.29	0.64	0	0	0.07	0	0
3	0	0	0.28	0.71	0	0	0	0
4	0	0.07	0	0	0.50	0.14	0	0.28
5	0	0.36	0	0	0	0.57	0	0.07
6	0	0.64	0	0	0.07	0.28	0	0
7	0	0.07	0	0	0.14	0.21	0	0.57

FIGURE 10. Results from clustering 2-D synthetic data, as depicted in Fig. 1(c), using (a) amplitude embedding and (b) angle embedding on hardware running circuits for 8192 shots on IBMQ Sydney.

cluster, as seen in Fig. 1(c). The number of data points and clusters was selected to most easily work with the `Circuit Runner` service, reaching the limits to how many circuits can be sent at once.

In the case of the four cluster *easy* and *hard* datasets, we analyze the quantum clustering approach using the same execution parameters for both datasets. We use both amplitude and angle embedding with a maximum of five iterations or until convergence of the centroid locations is reached. We use the outputs of the classical algorithm implementation as the base truth, 8192 shots for each experiment, and use the option for measurement error mitigation in all cases. For these sets of experiments, we submit jobs to quantum devices that have a quantum volume of 32. The results of the experiments for 2-D are seen in Fig. 9. The results displayed are, on the left side, left column, the confusion matrices comparing the classical baseline to the quantum outputs using amplitude embedding, and the equivalent for angle embedding in the right column. We observe that in the simulation setting, the classical outputs are matched perfectly; however, for real hardware, the results show a relatively low accuracy in the labeling for both embeddings. The results are similar to the *hard* dataset, as seen on the right side of Fig. 9.

For the 8-cluster data, we perform the same simulation steps but for a maximum of three iterations with 8192 shots. In simulation, convergence is reached with two iterations with perfect labeling results. We test both 2-D and 4-D datasets. The 2-D results are shown in Fig. 10. For 4-D data, the results had very low accuracy with essentially a random labeling and we neglect to show the results here.

Overall, these experiments motivate that using one instance of executing the distance estimation circuit with low shot counts produces a distance estimation that is not accurate enough to cluster data. To mitigate noise, we predictthat we can instead average multiple distance estimations to improve estimation consistency while simultaneously increasing the circuit shots. We test this hypothesis in Section III-C.

C. SUMMARY OF BENCHMARKING RESULTS

In summary, we benchmarked three key points for performing distance estimation and clustering on a quantum computer. First, for distance estimation quality, we benchmarked distance estimation accuracy versus the total number of shots used. We find, generally, a high number of shots—larger than 10 000— should be used to get fairly consistent results. Using values less than 10 000, we find that the noise levels are too high to provide high accuracy

and consistency. Next, we considered how increasing the Euclidean distance affects accuracy. Here, we found that with amplitude embedding, the results trended more similarly to the noiseless simulated results, whereas the (dense) angle embedding approach tended to diverge from the simulated value. In our last experiment, we investigated how the dimension of the data affects the distance estimation for increasing dimension size. We found that with amplitude embedding, the simulation results matched more closely to the hardware, but with angle embedding, the values tended to diverge more quickly. We believe this is due to both the additional number of controlled swaps required as well as the finite precision of the physical qubit rotation gates.

Building upon these distance-related metric results, we benchmarked the ability to perform clustering with synthetic data. We tested clustering in terms of varying the dimension of the data, the cluster tightness, and in the number of total clusters. We found that the real hardware performed best in the easiest cases with only 2-D data. By increasing the number of clusters, we expectedly observed a sharp fall in accuracy using our parameter regime.

In the field of quantum computing, being able to produce accurate and consistent results using noisy quantum computers is a major challenge. Here, we have conducted an initial study of accuracy, but future work involves finding the best parameter settings, such as data dimension and well-clusterable data types, that can provide both good performances in terms of runtime and accuracy. In Section IV, we propose a scheme that can break down larger-scale problems into smaller problems that are within such a parameter regime.

IV. APPLICATIONS FOR ENERGY SUBGRID CLUSTERING

The motivation for performing a detailed analysis on how well real quantum hardware performs on various distance-related metrics for high-dimensional data is because, typically, many real-world datasets are nontrivial, high dimensional, and not well clusterable. In classical computing solutions involving unsupervised clustering, the problem of aggregating and grouping sets of objects usually involves some dimension reduction techniques and some type of domain knowledge to tune any machine learning algorithm used. Indeed, a standard k-means approach—in the classical sense—when used in practice involves a fine balance of mitigation and optimization techniques depending on the data. Selecting the optimal clusters, choosing the initial centroid locations, choosing an optimal subset size of the dataset for large datasets, dealing with outliers, and dimension reduction are all points to consider when applying k-means to an unlabeled dataset. Celebi et al. analyze the effects of these points in [26].

Here, we focus more on the quantum aspect of k-means clustering. After using standard techniques for preprocessing the data and selecting initial centroids, we direct our efforts to improve the distance estimation accuracy. Since quantum

TABLE 3 Description of the Features for Each Powerline in the Dataset

Name	Unit
Conductor cross-section	cm^2
Operating Voltage	kV
Average Renewable Energy In-feed Load	MWh
Average Non-Renewable Energy In-feed Load	MWh
Number of Exits of Next Major Substation	#
Line Length	m
Sum MVA at closest HV exist	MVA

machine learning and variations of clustering are touted as being a possible avenue for quantum advantage, we aim to employ the findings in the previous sections in a real use-case that provide business value. In doing so, we propose an alternative implementation of the distance estimation circuit to overcome some of the deficiencies revealed in the previous section, namely, the inability of the vanilla quantum k-means algorithm to be able to handle classical input vector dimensions more than four.

A. GERMAN ELECTRICITY GRID DATA

Predictive maintenance is a major area of applied research in the energy operations sector [27], [28]. The ability to determine areas of the electrical grid which are susceptible to failing in some predetermined timespan has many obvious benefits for customers downstream from any grid infrastructure which may fail. One possible approach to this problem is using data-driven analysis of different partitions of the full network grid to group and find similar types of subgrid assets. This can be done by taking into account data features such as 1) the amount of renewable and nonrenewable electricity flowing through the grid subsection; 2) the number of power lines within a subsection; and 3) descriptive statistics about the ages of the assets contained in the subsection. Given such a collection of asset properties for electrical grid assets, we aim to employ unsupervised k-means clustering to classify various subgrids of part of the German Electrical Grid [29].

The dataset consists of 81 350 low-voltage power lines from a distribution system operator grid in Germany. Each power line has seven numerical features, as described in Table 3. Low-voltage subgrid networks are connected to high-voltage entry and exit points in the grid. For a given high-voltage transformer in the grid network, we collect the low-voltage lines that are part of its respective subnetwork and compute numerical features describing the entire subset of low power lines. Specifically, for each subgrid we compute: the number of nonpowerline assets, the total number of connected assets, and the minimum, maximum, and sum of each of the features are listed in Table 3. There are 1037 subgrids and, therefore, we have a final dataset of 1037 feature vectors, each of dimension 26.

B. RESULTS

To cluster the 26-D data created using the individual power line features (see Table 3) for all power lines in a given high voltage entry point, we first perform a preprocessing step to reduce the total feature vector dimension. In order to fit the data onto the quantum computers available for this work, we reduce the dimension to eight using principal component analysis (PCA) which results in a 97.7% explained variance as well as a dataset with six dimensions using PCA, which accounted for 91.4% of the variance. This second dataset was used for the angle embedding approach to fit in a 7-qubit quantum computer, the quantum computer topology we had the most access to in this work.

With an initial classical analysis using the elbow method [30], the optimal number of clusters for this dataset was determined to be $k = 5$. From this dimension-reduced dataset of 1087 points, we randomly selected 180 points to cluster, where 180 points allow us to send 900 circuits $(180 \cdot 5 = 900)$ to IBM's cloud service in one job (an upper limit for some hardware). Important to any unsupervised clustering algorithm is the choice of initial centroid points. In order to ensure a quick convergence, and to reduce the number of quantum iterations, we ran the classical algorithm with a variety of random seeds such that convergence was reached within three iterations. The classical clustering results are depicted in Fig. 11(a), using t-distributed stochastic neighbor embedding (t-SNE) [31] on the high-dimensional data to generate a 2-D projection. Using the initial centroids that achieved this, we then ran the quantum clustering experiment.

To validate the quantum approaches we used, we first compare the labeling output from noiseless simulation to the label output using the classical approach and then repeat the comparison running on real quantum devices. Because the classical approach converged in three iterations, we allow the quantum versions to run with a maximum of five iterations. In simulation, the balanced accuracy of the experiments was 100% with amplitude embedding and 97.8% for angle embedding. Given that simulation produces high accuracy, we performed a series of tests on the quantum hardware.

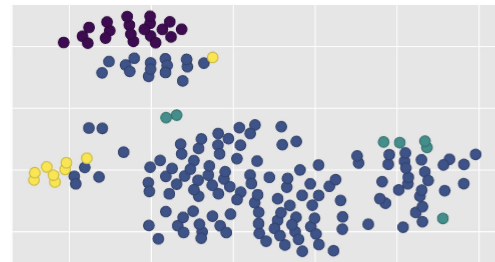### 1) CLUSTERING USING AMPLITUDE EMBEDDING

The first test we perform is to simply run the same logic as in the simulation. We use 12 000 shots per distance estimate and run the full clustering algorithm for five iterations. The clustering result using amplitude embedding is given in Fig. 11(b). The grid data are of relatively high dimension and the circuits to prepare the data are roughly 120 gate depth with approximately 70 nonlocal gates for amplitude embedding. For angle embedding, the gate depth is expectantly shallower at approximately 86 but with roughly 103 nonlocal gates, depending on the randomization of the circuit transpilation step. With the level of noise occurring, five iterations do not improve the results, and indeed, we speculate further iterations would not have led to improved results either. Here, our observation is that the labeling is essentially random due
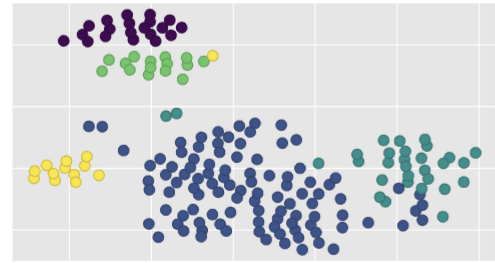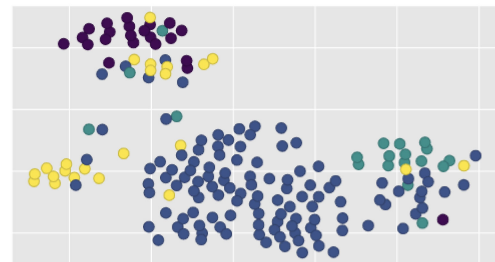

(a)
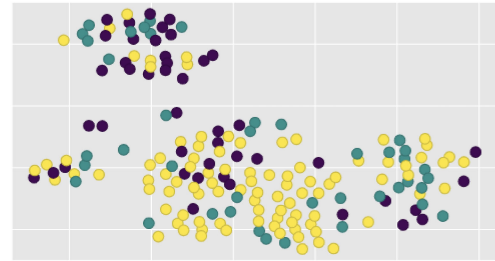
(b)

(c)

(d)

FIGURE 11. (a) Classical clustering output after t-SNE is performed on the subgrid data. (b) Quantum labeling output using 8-D amplitude embedding for five iterations with 12 000 shots per distance estimation on IBM Casablanca. (c) Quantum clustering output using the angle embedding split distance estimation (12) of the subgrid data. We decomposed the distance estimation to be one estimate per circuit, for five iterations with 12 000 shots per distance estimation on IBM Perth. (d) Quantum clustering using amplitude embedding split distance estimation (12) using a parallel execution process as described in (c).

to the noise in the distance estimation circuits, never leading to a converging state.

2) CLASSIFICATION USING AMPLITUDE EMBEDDING

As a second test, we implemented a pure nearest-neighbor classification application. We begin by training the model
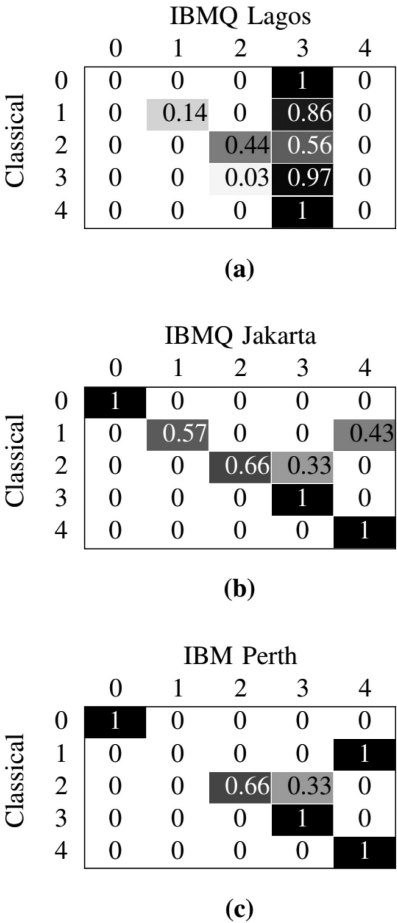
IBMQ Lagos

	0	1	2	3	4
0	0	0	0	1	0
1	0	0.14	0	0.86	0
2	0	0	0.44	0.56	0
3	0	0	0.03	0.97	0
4	0	0	0	1	0

(a)

IBMQ Jakarta

	0	1	2	3	4
0	1	0	0	0	0
1	0	0.57	0	0	0.43
2	0	0	0.66	0.33	0
3	0	0	0	1	0
4	0	0	0	0	1

(b)

IBM Perth

	0	1	2	3	4
0	1	0	0	0	0
1	0	0	0	0	1
2	0	0	0.66	0.33	0
3	0	0	0	1	0
4	0	0	0	0	1

(c)

FIGURE 12. Results of classifying the test set of 60 data points using a nearest-neighbor prediction. (a) Amplitude embedding in eight dimensions and 30 000 shots per distance estimation. The balanced accuracy in this case is 17.0% and the raw accuracy 46.7%. Weighted precision cannot be computed since some classes are empty. (b) Repeat the classification using 15 000 shots using the divided distance estimation in four 2-D estimates, using an average of five estimates per distance. The results have a balanced accuracy of 84.8%, a raw accuracy of 90.0%, and a weighted precision score of 92.1%. (c) Classification is done using the vector subspace parallelization circuit that uses two swap-tests per circuit. This approach reduces the total number of circuits by 50%. The results improved significantly to have a balanced accuracy of 73.3% and a raw accuracy of 83.3%.

offline classically to determine optimal centroid locations, then, at runtime, we compute only the prediction step quantumly to determine which cluster test set data points belong to. Fig. 12(a) shows the accuracy results of the outcome, where we used 30 000 shots to estimate the distances using amplitude embedding in 8-D. We see that the majority of points were assigned to one class, similarly to how the five iterations of clustering performed.

3) DISTANCE ESTIMATION WITH VECTOR SUBSPACE PARALLELIZATION

So far, our approach to distance estimation has been to use the approach, as stated in [5]; however, the accuracy in practice using this approach has thus far been relatively low. From the benchmarking section, the highest accuracy was seen in the

2-D data experiments. Using this as motivation, we propose a new technique of parallelizing the distance calculation for high-dimensional vectors by using distances between 2-D subspaces of the full feature vectors. In classical clustering problems, for large dimensional datasets, subspace clustering has been considered to cluster subspaces of the data [32]. In our case, we can use approaches for large dimensional datasets, but for small datasets, accommodating near-term quantum computers. We consider the entire space as a sum of the subspaces rather than find clusters in any particular subspaces, as described in [32].

Given input data vectors $a := (a_1, a_2, \ldots, a_n)$ and $b := (b_1, b_2, \ldots, b_n)$, the distance between them can be decomposed as

$$
\begin{aligned}
d(a, b) = \; & d(a_{1,2}, b_{1,2}) + d(a_{3,4}, b_{3,4}) \\
& + \cdots + d(a_{n-1,n}, b_{n-1,n})
\end{aligned}
\tag{12}
$$

where $a_{i,j} = \mathcal{P}_{i,j}(a)$ and $b_{i,j} = \mathcal{P}_{i,j}(b)$ are projections of the respective vectors to the (i, j)th vector subspace. The circuit for this parallel distance estimate using angle embedding is depicted in Fig. 13(a) and amplitude embedding in (b). One can extend this approach to any form of 2-D data embedding, filling a quantum processor with many simultaneous distance estimates. Moreover, as quantum computers improve, more dimensions can be included in the decomposition, for example, using 4-D decompositions instead of 2-D. When filling multiple circuits into one quantum processor, we need to be concerned about the cross talk [33], and future work will be to investigate optimal qubit layouts to maximize performance.

This approach has various benefits in terms of mitigating noise. First, it uses only low-dimensional projections. In this case, we use 2-D projections, aligned with our benchmarking results, but as hardware improves, we can extend this to larger dimensions to reduce the number of total independent measurements until we can eventually use the entire vector. Next, these low-dimensional circuits will be, in general, shallower and thinner, which will improve the accuracy and moreover reduce the computation time, allowing for more shots within the same execution timespan. Because in some cases we observed a large standard deviation, with a shorter execution time, one can also execute the circuit many times to produce an average distance estimate the same timespan, mitigating Gaussian noise in the system.

Because the distance estimation circuits are thinner, we can load multiple circuits into one quantum processor proportional to the number of qubits. For example, because the swap-test with angle embedding in 2-D uses three qubits per swap-test, we can load two distance calculations at a time in a seven qubit quantum computer, reducing the number of total circuits to execute by 50%. This approach could be generalized to contain as many swap-tests as there are (the floor of) one-third the number of qubits, which could in turn result in again using one circuit for distance estimation, simply with a modified pre- and postprocessing step. These benefits
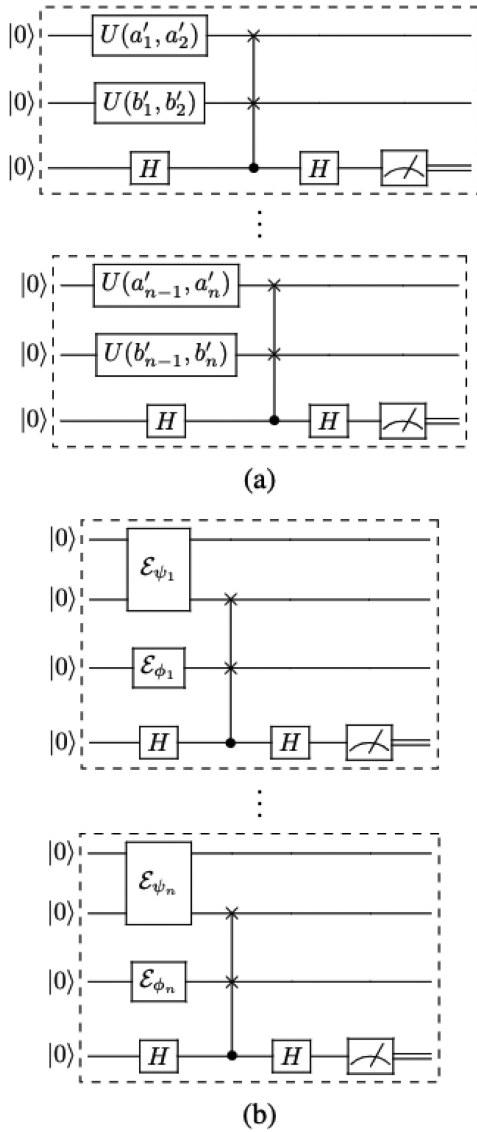
FIGURE 13. Circuit for processing multiple distances in one circuit using (a) angle embedding and (b) amplitude embedding with swap tests. The dashed boxes separate each independent swap-test.

make this much more NISQ compatible than performing the distance estimation with all dimensions considered at once.

After verifying this approach produced accurate results in simulation, we tested how well it mitigates the effects of noise in the classification task, we ran the circuit implementing (12) using two approaches. For the first approach, we used amplitude embedding and executed each distance estimation circuit independently five times, averaging the results and using the average as the distance estimate. We used 15 000 shots per execution and since the circuit uses four qubits, we could fit just one circuit at a time on the 7-qubit device. The confusion matrix of the results is in Fig. 12(b), showing a vast improvement. The balanced accuracy in this case is 84.8% compared to 17% using the standard quantum k-means algorithm, an improvement of 67.8%.

For the second approach, we perform classification again, now using angle embedding, but in this case, since just three qubits per swap-test are required, we loaded two distance estimates in parallel into the 7-qubit device. Using 15 000 shots with, in this case, two repetitions per circuit, we again use the average for the estimate. The results of the classification are shown in Fig. 12(c). Again, we see a strong improvement for the classification problem over the standard method of embedding all the data at once and using a larger number of controlled swaps, increasing the balanced accuracy from 17% in the standard approach to 73.3% using this novel approach, a difference of 56.3%.

4) CLUSTERING WITH VECTOR SUBSPACE PARALLELIZATION

Given the promising results from the classification task using the vector subspace parallelization, we again perform the full clustering algorithm using the angle embedding approach. We use the distance estimation (12) with 18 000 shots and one measurement repetition on the IBM Perth machine with a total of five iterations for the clustering algorithm. The clustering results are shown in Fig. 11(c). Although the labels were reduced to four classes, one fewer than in the classical algorithm, we see a much clearer separation of the classes in comparison to using amplitude embedding processing all eight dimensions in the standard approach. The balanced accuracy compared to the classical label results from clustering in this case is 58.5%. We use the same subspace approach with 8-qubits on the IBM Montreal machine, using two amplitude encoded circuits, as in Fig. 11(d), with 12 000 shots and balanced accuracy of 63.9% in comparison to the classical clustering label results. Overall, these results show a vast improvement to the standard approach which would consider all dimensions in one measurement.

V. CONCLUSION AND OUTLOOK
In this work, we thoroughly investigated the potential of using quantum k-means clustering in a practical manner on current NISQ quantum hardware. In terms of distance estimation comparison between classical and quantum distance calculations, we clearly observed a high level of difference between simulation and running on physical devices—especially comparing the distance estimation results using angle embeddings. The results which used batched job submission via Qiskit Runtime showed vastly improved performance, allowing for more circuit executions, improving reliability, as well as drastic speed improvements when dealing with large datasets due to the reduction in job-queuing time.

The best k-means clustering results observed were from clustering datasets of 2-D data points. When we increased the number of clusters from four to a more complex scenario of eight clusters and changed the input vector dimension from two to four, the results worsened. We experimented with an industrial unsupervised learning problem, labeling high-dimensional energy grid data using k-means clustering. Using the standard approaches, the clustering and classification

results proved inaccurate when executed over real hardware. When we changed the distance metric and used our vector subspace parallelization approach, we saw a significant improvement in both our classification and clustering experiments. For amplitude embeddings, the balanced accuracy of the classification went from 17% using the standard approach to 84.8% with this novel distance estimation method. With angle embedding, loading two swap-tests into one circuit to execute in parallel, albeit an overall wider circuit, proved to also have a large performance improvement over the amplitude embedding approach with a balanced classification accuracy of 73.3% and raw accuracy of 83.3%.

This work provides the first step into quantum clustering for practical industrial use-cases but still there are questions to be answered. Future work will be to consider other clustering algorithms, such as *k*-medoids which uses alternative distance metrics, considering other quantum approaches for distance estimation as in [14] and [34] or those better suited for NISQ hardware as proposed in [35]. Indeed, many algorithms require a distance calculation step, and so benchmarking their quantum performance leaves many possibilities for future work.

Although it is well known that quantum computing is in its early stages of development, it is important to investigate what boundaries exist in relation to nontrivial problems that move beyond fundamental algorithm proof-of-concepts. Clustering and, particularly, distance estimation are widely used in various industrial applications. With this work, we have tested a large set of experiments that can be performed on the quantum cloud using only the core features of the platform. Even though we do not believe in the current stage, using quantum computers for clustering will add a benefit, quantum technology is continuously and rapidly improving. With this, we expect that as NISQ-era quantum computers mature, these types of analysis and industry-driven use-case studies will continue to provide valuable insight into how they will be used for real-life applications.

ACKNOWLEDGMENT

REFERENCES

[1] C. G. Almudever et al., "The engineering challenges in quantum computing," in *Proc. Des., Automat. Test Europe Conf. Exhib.*, 2017, pp. 836–845, doi: 10.23919/DATE.2017.7927104.

[2] "Quantum services," Accessed: Aug. 1, 2022. [Online]. Available: https://quantum-computing.ibm.com/services

[3] S. Lloyd, "Least squares quantization in PCM," *IEEE Trans. Inf. theory*, vol. 28, no. 2, pp. 129–137, Mar. 1982, doi: 10.1109/TIT.1982.1056489.

[4] M. Schuld and F. Petruccione, *Supervised Learning With Quantum Computers*, vol. 17. Berlin, Germany: Springer, 2018, doi: 10.1007/978-3-319-96424-9.

[5] S. Lloyd, M. Mohseni, and P. Rebentrost, "Quantum algorithms for supervised and unsupervised machine learning," 2013, doi: 10.48550/arXiv.1307.0411.

[6] H. Buhrman, R. Cleve, J. Watrous, and R. De Wolf, "quantum fingerprinting," *Phys. Rev. Lett.*, vol. 87, no. 16, 2001, Art. no. 167902. doi: 10.1103/PhysRevLett.87.167902.

[7] L. K. Grover, "A fast quantum mechanical algorithm for database search," in *Proc. 29th Annu. ACM Symp. Theory Comput.*, 1996, pp. 212–219, doi: 10.48550/arXiv.quant-ph/9605043.

[8] M. Plesch and I. C. V. Brukner, "Quantum-state preparation with universal gate decompositions," *Phys. Rev. A*, vol. 83, Mar. 2011, Art. no. 032302, doi: 10.1103/PhysRevA.83.032302.

[9] E. Tang, "Quantum principal component analysis only achieves an exponential speedup because of its state preparation assumptions," *Phys. Rev. Lett.*, vol. 127, no. 6, 2021, Art. no. 060503, doi: 10.1103/PhysRevLett.127.060503.

[10] S. U. Khan, A. J. Awan, and G. Vall-Llosera, "K-means clustering on noisy intermediate scale quantum computers,"2019, *arXiv.1909.12183*.

[11] S. Johri et al., "Nearest centroid classification on a trapped ion quantum computer," *NPJ Quantum Inf.*, vol. 7, no. 1, pp. 1–11, Aug. 2021, doi: 10.1038/s41534-021-00456-5.

[12] V. Kumar, G. Bass, C. Tomlin, and J. Dulny, "Quantum annealing for combinatorial clustering," *Quantum Inf. Process.*, vol. 17, no. 2, pp. 1–14, 2018, doi: 10.1007/s11128-017-1809-2.

[13] D. Arthur et al., "Balanced k-means clustering on an adiabatic quantum computer," *Quantum Inf. Process.*, vol. 20, no. 9, pp. 1–30, 2021, doi: 10.1007/s11128-021-03240-8.

[14] K. Benlamine, Y. Bennani, A. Zaiou, M. Hibti, B. Matei, and N. Grozavu, "Distance estimation for quantum prototypes based clustering," in *Proc. Int. Conf. Neural Inf. Process.*, Cham, Switzerland: Springer, 2019, pp. 561–572, doi: 10.1007/978-3-030-36718-3_47.

[15] K. Benlamine, Y. Bennani, N. Grozavu, and B. Matei, "Quantum collaborative k-means," in *Proc. Int. Joint Conf. Neural Netw.*, 2020, pp. 1–7, doi: 10.1109/IJCNN48605.2020.9207334.

[16] I. Kerenidis, J. Landman, A. Luongo, and A. Prakash, "q-means: A quantum algorithm for unsupervised machine learning," in *Advances in Neural Information Processing Systems*, H. H. Wallach, A. Larochelle Beygelzimer, F. dAlché-Buc, E. Fox, and R. Garnett, Eds., vol. 32. Red Hook, NY, USA: Curran Associates, 2019. doi: 10.5555/3454287.3454659. [Online]. Available: https://dl.acm.org/doi/abs/10.5555/3454287.3454659

[17] C.-H. Nguyen, K.-W. Tseng, G. Maslennikov, H. Gan, and D. Matsukevich, "Experimental swap test of infinite dimensional quantum states," 2021, *arXiv.2103.10219*.

[18] Qiskit, "Initialize," Accessed: May 18, 2022. [Online]. Available: https://qiskit.org/documentation/stubs/qiskit.extensions.Initialize.html

[19] R. LaRose and B. Coyle, "Robust data encodings for quantum classifiers," *Phys. Rev. A*, vol. 102, no. 3, 2020, Art. no. 032420, doi: 10.1103/PhysRevA.102.032420.

[20] D. Kopczyk, "Quantum machine learning for data scientists," 2018, *arXiv.1804.10068*.

[21] E. Fredkin and T. Toffoli, "Conservative logic," *Int. J. Theor. Phys.*, vol. 21, no. 3, pp. 219–253, 1982, doi: 10.1007/BF01857727.

[22] A. Cross, "The IBM Q experience and qiskit open-source quantum computing software," in *Proc. APS Mar. Meeting Abstr.*, Mar. 2018, pp. L58–003.

[23] R. Parekh, A. Ricciardi, A. Darwish, and S. DiAdamo, "Quantum algorithms and simulation for parallel and distributed quantum computing," in *Proc. IEEE/ACM 2nd Int. Workshop Quantum Comput. Softw.*, 2021, pp. 9–19, doi: 10.1109/QCS54837.2021.00005.

[24] Qiskit, "More circuit identities," Accessed: May 18, 2022. [Online]. Available: https://qiskit.org/textbook/ch-gates/more-circuit-identities.html#5.-Arbitrary-rotations-from-H-and-T.

[25] P. Murali, J. M. Baker, A. Javadi-Abhari, F. T. Chong, and M. Martonosi, "Noise-adaptive compiler mappings for noisy intermediate-scale quantum computers," in *Proc. 24th Int. Conf. Archit. Support Program. Lang. Oper. Syst.*, 2019, pp. 1015–1029, doi: 10.1145/3297858.3304075.

[26] M. E. Celebi, H. A. Kingravi, and P. A. Vela, "A comparative study of efficient initialization methods for the k-means clustering algorithm," *Expert Syst. Appl.*, vol. 40, no. 1, pp. 200–210, 2013, doi: 10.1016/j.eswa.2012.07.021.

[27] R. Fioravanti, K. Kumar, S. Nakata, B. Chalamala, and Y. Preger, "Predictive-maintenance practices: For operational safety of battery energy storage systems," *IEEE Power Energy Mag.*, vol. 18, no. 6, pp. 86–97, Nov./Dec. 2020, doi: 10.1109/MPE.2020.3014542.

[28] M. A. Mahmoud, N. R. Md Nasir, M. Gurunathan, P. Raj, and S. A. Mostafa, "The current state of the art in research on predictive maintenance in smart grid distribution network: Fault's types, causes, and prediction methods—A systematic review," *Energies*, vol. 14, no. 16, 2021, Art. no. 5078, doi: 10.3390/en14165078.

[29] T. P. Carvalho, F. A. Soares, R. Vita, R. d. P. Francisco, J. P. Basto, and S. G. Alcalá, "A systematic literature review of machine learning methods applied to predictive maintenance," *Comput. Ind. Eng.*, vol. 137, 2019, Art. no. 106024, doi: 10.1016/j.cie.2019.106024.

[30] R. L. Thorndike, "Who belongs in the family?," *Psychometrika*, vol. 18, no. 4, pp. 267–276, 1953, doi: 10.1007/BF02289263.

[31] L. Van der Maaten and G. Hinton, "Visualizing data using t-SNE," *J. Mach. Learn. Res.*, vol. 9, no. 11, pp. 2579–2605, 2008. https://jmlr.csail.mit.edu/papers/v9/vandermaaten08a.html

[32] L. Parsons, E. Haque, and H. Liu, "Subspace clustering for high dimensional data: A review," *ACM SIGKDD Explorations Newslett.*, vol. 6, no. 1, pp. 90–105, 2004, doi: 10.1145/1007730.1007731.

[33] Y. Ohkura, T. Satoh, and R. Van Meter, "Simultaneous execution of quantum circuits on current and near-future NISQ systems," *IEEE Trans. Quantum Eng.*, vol. 3, 2022, Art. no. 2500210, doi: 10.1109/TQE.2022.3164716.

[34] A. R. Kuzmak, "Measuring distance between quantum states on a quantum computer," *Quantum Inf. Process.*, vol. 20, no. 8, pp. 1–18, Aug. 2021, doi: 10.1007/s11128-021-03196-9.

[35] M. Fanizza, M. Rosati, M. Skotiniotis, J. Calsamiglia, and V. Giovannetti, "Beyond the swap test: Optimal estimation of quantum state overlap," *Phys. Rev. Lett.*, vol. 124, no. 6, 2020, Art. no. 060503, doi: 10.1103/PhysRevLett.124.060503.