Quantum Machine Learning for Controller Placement in Software Defined Networks

Swaraj Shekhar Nande$^{1,2}$, Osel Lhamo$^1$, Marius Paul$^{1,2}$, Riccardo Bassoli$^{1,2,3}$, and Frank H P Fitzek$^{1,3}$

$^1$Deutsche Telekom Chair of Communication Networks, Technische Universität Dresden
$^2$Quantum Communication Networks (QCNets) research group, Technische Universität Dresden
$^3$Cluster of Excellence, Centre for Tactile Internet with Human-in-the-Loop (CeTI)

February 27, 2024
Quantum Machine Learning for Controller Placement in Software Defined Networks

Swaraj Shekhar Nande∗†, Osel Lhamo∗, Marius Paul∗†‡, Riccardo Bassoli∗†‡ and Frank H. P. Fitzek∗†

∗ Deutsche Telekom Chair of Communication Networks, Technische Universität Dresden, Dresden, Germany.
† Centre for Tactile Internet with Human-in-the-Loop (CeTI), Cluster of Excellence, Dresden, Germany.
‡ Quantum Communication Networks (QCNets) research group, Technische Universität Dresden, Dresden, Germany.

E-mails: {swaraj_shekhar.nande, osel.lhamo, marius.paul, riccardo.bassoli, frank.fitzek}@tu-dresden.de

Abstract—Future 6G networks will be enabled by full softwarization of network functions & operations and in-network intelligence for self-management and orchestration. However, the intelligent management of a softwarized network will require massive data mining, analytics, and processing. That is why it is fundamental to find additional resources like quantum technologies to help achieve 6G key performance indicators. Quantum properties provide quantum computers to run a quantum algorithm with lesser queries. Quantum Machine Learning (QML) studies machine learning techniques on quantum computers. In this work, we use a QML algorithm to solve the controller placement problem for a multi-controller Software Defined Network (SDN). The network delay depends on where the controller is located, thus, it is critical to choose controllers at positions leading to minimize latency between the controllers and their associated switches. We consider an SDN architecture which is in its early stage of installation where the network nodes are deployed but connections will be established after obtaining controller locations, which results in the reduction of the overall controller to switch delay. By using different types of datasets, i.e., uniformly distributed and Gaussian distributed points, the experimental results show that the QML algorithm accelerates the SDN clustering methods (which are used to resolve the control placement problem) compared to those of the classical machine learning algorithm (like K-means) with comparable latency.

Index Terms—Quantum Computing, SDN, K-means cluster, Controller Placement.

I. INTRODUCTION

With the ever-changing and developing scope of the network infrastructure, it is challenging to manage them using conventional means such as manual configurations since these manual methods are vulnerable to human errors, scalability issues etc. Software Defined Network (SDN) decouples control and data plane, which allows dedicated controller instances to manage network devices by defining packet forwarding rules, in order to efficiently react to quick changing traffic. SDN is establishing itself more firmly in the networking industry, as evident from the growing number of networking tools and SDN-related software deployed.

However, determining optimal placement of controllers for a given network framework is one of the long-standing issues of SDN. Generally, the objective of such a problem is to increase the network performance in terms of latency, reliability, energy conservation and load balancing. These different network parameters to be optimized often conflict with each other such that one metric must be compromised to improve another. Hence, controller placement may have to include an appropriate trade-off between the metrics pertinent to a specific use case.

As demonstrated in [1], the controller placement can be formulated as a facility location problem that is NP-hard and usually emerges when optimizing the location of warehouses and factories, among other situations. The authors observed that typically one controller is sufficient to satisfy the latency requirements of medium size networks but is not enough to ensure fault tolerance of production networks. Therefore, to efficiently operate large-scale networks, multiple controllers are required to avoid a single point of failure.

Similar to works in [2], [3] and [4], we use a clustering algorithm to divide the network into sub-networks (clusters). With the machine learning clustering techniques like K-means, the maximum latency between the controller (centroid) and the related switches (nodes) of that cluster can be shortened, and each sub-network is managed by one SDN controller. It is notable that as the amount of data generated in sub-networks are expected to grow faster than the growth in its computational capabilities, this leads to the need of more powerful ways of processing information.

Quantum computation uses fundamental properties of quantum physics to redefine how computers create and manipulate information. This field studies the quantum behaviour of certain subatomic particles (photons, electrons, etc.) for subsequent use in performing calculations and large-scale information processing. Quantum computers provide a radically new way of computing by using qubits instead of bits and give the possibility of obtaining quantum algorithms that could be substantially faster than classical algorithms. These advantages can be achieved through quantum features, such as entanglement or superposition. These capabilities can give quantum computers an advantage in terms of computational time and cost over classical computers. Among different quantum algorithms, QML algorithms are one of the most promising applications of a full-scale quantum computer. In
recent years, there have been proposals for quantum machine learning algorithms that can offer considerable speedups over the corresponding classical algorithm [5]. While methods such as $K$-means have already been implemented for controller placement in SDN, our work aims to explore the possibility of improving $K$-means clustering by applying QML algorithms for clustering in SDN applications. Specifically, we would be using the $q$-means algorithm with convergence and precision guarantees similar to $K$-means, and it outputs with high probability a good approximation of the $K$ cluster centroids like the classical algorithm [6]. $q$-means running time is polynomial in the number of data points $N$; this provides substantial savings compared to the classical $K$-means algorithm that has a linear time complexity.

We propose to use the quantum algorithm on a quantum computer to determine placement of controllers for a large SDN architecture. This architecture would be consisting of multiple switches (deployed at different geographical position) where the wired link has not been established yet. The quantum algorithm is used for dividing the whole network and clustering them into several sub-networks each of which having a SDN controller at its centroid. These centroids (controllers) are determined to decrease controller-to-switch response time for any sub-network (cluster).

Following is the structure of the remainder of this paper. Section II briefly discusses the theoretical concepts that forms the basis of this work. Section III covers the work related to our research. We present our solution in section IV. Section V evaluates our proposed solution. Lastly, section VI summarizes our paper and observes possible future work.

II. BACKGROUND

We briefly explain the relevant theoretical concepts in this section to better understand our approach. First, we define SDN architecture and the importance of controller placement problems. Then we discuss the $K$-means clustering method. And finally, provide the background knowledge necessary for quantum clustering.

A. Software Defined Network

Until recently, a conventional network infrastructure consists of network devices that tend to be totally locked down with fixed hardware and software. With the increasingly dynamic nature of today’s applications, it became imperative to introduce programmability in the network to support them. This resulted in the emergence of SDN, which has an architecture as layers that provides a separate control and data plane, enabling network programmability in a central manner. Since SDN is a software-based network, users can manage resources virtually via the control plane, which allows resources to be provisioned from a single point in the network. As the blooming real-time applications and the growing network scale, multi-controller SDN architectures started being suggested, leading to the need for an optimum controller deployment strategy. As shown in Fig. 1, a multi-controller SDN architecture consists of several sub-networks, each managed by a controller. Determining the controllers’ positions in such a architecture is critical to reduce latency between the controller and the switches in its sub-network.

![Fig. 1. An example of multi-controller SDN architecture which is clustered into three sub-networks managed by one controller each.](image)

B. $K$-means cluster

The $K$-means algorithm was introduced in [7] and is extensively used for unsupervised problems. The inputs to the $K$-means algorithm are vectors $v_i \in \mathbb{R}^d$ for $i \in [N]$, where $[N]$ represents the numbers of vectors in the dataset. These points must be partitioned in $K$ subsets according to a similarity measure, which in $K$-means is the Euclidean distance between points. The output of the $K$-means algorithm is a list of $K$ cluster centres, which are called centroids. The algorithm starts by selecting $K$ initial centroids randomly or using efficient heuristics like the $K$-means++ [8]. It then alternates between two steps: (i) Each data point is assigned the label of the closest centroid. (ii) Each centroid is updated to be the average of the data points assigned to the corresponding cluster. These two steps are repeated until convergence, that is, until the change in the centroids during one iteration is sufficiently small. More precisely, we are given a dataset $V$ of vectors $v_i \in \mathbb{R}^d$ for $i \in [N]$. At step t, we denote the $K$ clusters by the sets $C_j^t$ for $j \in \{1, 2, \ldots, K\}$, and each corresponding centroid by the vector $c_j^t$. At each iteration, the data points $v_i$ are assigned to a cluster $C_j^t$ such that $C_1^t \cup C_2^t \cup C_K^t = V$ and $C_i^t \cap C_j^t = \emptyset$ for $i \neq j$. Let $d(v_i, c_j^t)$ be the Euclidean distance between vectors $v_i$ and $c_j^t$. The first step of the algorithm assigns each $v_i$ a label $l(v_i)^t$ corresponding to the closest centroid, that is

$$l(v_i) = \text{argmin}_{j \in \{1, 2, \ldots, K\}} (d(v_i, c_j^t))$$

(1)

The centroids are then updated, $c_j^{t+1} = \frac{1}{|C_j^t|} \sum_{i \in C_j^t} v_i$, so that the new centroid is the average of all points that have been assigned to the cluster in this iteration. We say that we have converged if for a small threshold $\tau$, we have

$$\frac{1}{K} \sum_{j=1}^{K} d(c_j^{t}, c_j^{t-1}) \leq \tau$$

(2)

This algorithm aims to minimize the loss function of the RSS (residual sums of squares), the sum of the squared distances between points and the centroid of their cluster. The RSS
decreases at each iteration of the K-means algorithm, and the algorithm converges to a local minimum. The number of iterations \(T\) for convergence depends on the data and the number of clusters. A single iteration has the complexity of \(O(kNd)\) since the \(N\) vectors of dimension \(d\) have to be compared to each \(K\) centroids.

\[
\text{RSS} := \sum_{j \in \{1,2,\ldots,K\}} \sum_{i \in C_j} d(c_j, v_i)^2
\]

(C. Quantum preliminaries)

In classical computing, any information is encoded using bits, where each bit can have the value zero or one. In quantum information science, the idea is to encode information using qubits. A qubit is a two-level quantum system where its two basis states are generally represented by \(|0\rangle, |1\rangle\). And it can either be in state \(|0\rangle \& |1\rangle\), or in a linear combination of both the states, which is fundamentally different from a classical concept. In superposition, quantum particles are a combination of all possible states. They fluctuate until they are observed and measured. Quantum interference is the intrinsic behaviour of a qubit, due to superposition, to influence the probability of it collapsing. Another extraordinary quantum property is Entanglement. Quantum particles can correlate their measurement results with each other. When qubits are entangled, they form a single system and influence each other. We can use the measurements from one qubit to conclude the others. By adding and entangling more qubits in a system on a quantum circuit, quantum computers can perform computations exponentially faster to solve more complicated problems. This introduces new concepts to traditional programming methods.

This work focuses on using the clustering and centroid estimation algorithm as proposed by Kerenidis et al. [6]. We would use the \(q\)-means algorithm to estimate the location of the controller’s placement in our SDN network architecture. Before starting, some quantum preliminaries needed for this work are:

1) Quantum Linear Algebra: A Quantum algorithm is expressed by linear algebra on a finite-dimensional complex inner product space. The mathematical formulations of quantum mechanics were established around 1930 by Von Neumann. The formulation uses functional analysis, linear algebra and probability theory. This section will discuss the linear algebraic formulation of quantum mechanics required to design the quantum circuits used in the \(q\)-means algorithm. For a symmetric matrix, \(M \in \mathbb{R}^{d \times d}\) with spectral norm \(||M|| = 1\) stored in the QRAM, the running time of these algorithms depends linearly on the condition number \(\kappa(M)\) of the matrix, that can be replaced by \(\kappa_s(M)\), a condition threshold where we keep only the singular values bigger than \(\tau\) and the parameter \(\mu(M)\), a matrix dependent parameter defined as

\[
\mu(M) = \min_{p \in [0,1]} \left(||M||_F, \sqrt{s_{2p}(M)(s_1-2p(M^T))}\right)
\]

for \(s_p(M) = \max_{i \in [n]} \sum_{j \in [d]} M_{ij}^p\). The different terms in the definition of \(\mu(M)\) correspond to different choices for the data structure for storing \(M\), as detailed in [6]. Note that \(\mu(M) \leq ||M||_F \sqrt{d} \leq ||M|| = 1\) has been assumed. Here running time also depends logarithmically on the relative error \(\epsilon\) of the final outcome state [9]. The linear algebra procedures above can also be applied to any rectangular matrix \(V \in \mathbb{R}^{N \times d}\) by considering instead the symmetric matrix

\[
\nabla = \begin{bmatrix} 0 & V^T \\ V & 0 \end{bmatrix}
\]

2) Quantum Random Access Memory: Many algorithms throughout different computing fields require memory storage and use. A ubiquitous form of memory storage in classical computing is random access memory (RAM). Similar to classical computing, many quantum computing algorithms also depend on the use of stored states. There are two main ways to achieve this. First is an algorithm that combines classical and quantum computing, only using quantum states for brief periods. Second, using QRAM [10]. QRAM uses the same three components as classical RAM: Memory Array, Input Register, and Output Register. QRAM works by using quantum superposition to perform memory access. To access a superposition of the memory cells the address register \(a\) must contain a superposition of the address. The QRAM returns a superposition of data in a data register \(d\) correlated to the address register. Let \(D_j\) be the content of the \(j^{th}\) memory cell. The work of a QRAM can be expressed as

\[
\sum_j \frac{1}{\sqrt{N}} |j_a\rangle |0\rangle_d \text{ QRAM } \sum_j \frac{1}{\sqrt{N}} |j_a\rangle |D_j\rangle_d
\]

3) Quantum Amplitude Estimation (QAE): When we perform some quantum operations on a quantum state, we generally prefer using the unitary operators. These unitary operations rotate the state vector in a multidimensional Hilbert space. Given an operator \(A\) that acts as

\[
A|0\rangle = \sqrt{1-a} |\Psi_0\rangle + \sqrt{a} |\Psi_1\rangle
\]

QAE is the task of finding an estimate for the amplitude \(a\) of the state \(\Psi_1\), i.e., \(a = |\langle \Psi_1 |\Psi_1\rangle|^2\), and where \(|\Psi_0\rangle\&|\Psi_1\rangle\) are orthogonal states. This was firstly investigated by Brassard et al. [11]. Their algorithm uses a combination of the Grover operator \(Q = AS_0A^\dagger S_{\Psi_1}\), where \(S_0\) and \(S_{\Psi_1}\) are reflections about the \(|0\rangle\) and \(|\Psi_1\rangle\) states, respectively, and phase estimation. For our case, given a quantum algorithm

\[
A : |0\rangle \rightarrow \sqrt{\tilde{p}}|y,1\rangle + \sqrt{1-\tilde{p}}|G,0\rangle
\]

where \(|G\rangle\) is some garbage state, then for any positive integer \(P\), the amplitude estimation algorithm outputs \(\tilde{p}(0 \leq \tilde{p} \leq 1)\) such that

\[
|\tilde{p} - p| \leq 2\pi\frac{\sqrt{p(1-p)}}{P} + \left(\frac{\pi}{P}\right)^2
\]

with probability at least \(\frac{8}{P}\). It uses exactly \(P\) iterations of the algorithm \(A\). If \(p = 0\) then \(\tilde{p} = 0\) with certainty, else if \(p = 1\) and \(P\) is even, then \(\tilde{p} = 1\) with certainty.
4) Quantum Median Evaluation: As we are working to solve the clustering problem and its centroid estimation for the SDN controller placement, the evaluation of the median of the quantum dataset is necessary. The quantum operation can be performed as such: let $u$ be a unitary operation that maps

$$ u : |0^{\otimes n}\rangle \rightarrow \sqrt{a} |x, 1\rangle + \sqrt{1-a} |G, 0\rangle $$

for some $1/2 < a < 1$ in time $T$. Then there exists a quantum algorithm that, for any $\delta > 0$ and for any $1/2 < a_0 < a$, produces a $|\Psi\rangle$ such that $|||\Psi\rangle - |0^{\otimes n}L\rangle| |x\rangle || \leq \sqrt{2\delta}$ for some integer $L$, in time [12]

$$ 2T \left[ \frac{\ln 1/\Delta}{2(|a_0| - \frac{1}{2})^2} \right] $$

5) Vector State Tomography: The final component of the $q$-means algorithm is a linear time algorithm for vector state tomography that will be used to recover classical information from the quantum state corresponding to the new centroids in each step [13]. Given a unitary $U$ that produces a quantum state $|x\rangle$, by calling $O(d \log d/d^2)$ times $U$, the tomography algorithm is able to reconstruct a vector $\pi$ that approximates $|x\rangle$ such that $||x\rangle - |x\rangle || \leq \epsilon$, where $\epsilon$ is the error parameter.

III. RELATED WORK

A. Controller Placement

The controller placement problem has been covered in many research papers, with one of the pioneer’s works being [1]. The author’s main objective was to optimize latency, particularly average and worst-case latency, using the k-centre algorithm. In [2], the authors use an optimized $K$-means algorithm to solve this problem and reduce the maximum latency between the centroid (controller) and their nodes compared to the simple $K$-means model. While the authors of [4] argue that using their proposed solution based on Fuzzy $C$-means approach offers a quicker response time than other $K$ clustering methods. We want to establish our solution based on the $K$-means algorithm to efficiently deploy controllers across our network but improve it by incorporating quantum science.

B. Quantum Clustering algorithms

Various researchers have made different approaches for implementing the clustering QML algorithm. [14] uses an improved quantum-inspired evolutionary fuzzy $C$-means (EQIE-FCM) algorithm to calculate the global optima of the parameters. The author uses “quantum $K$-means” and “quantum fuzzy $C$-means” clustering approaches to predict diabetes. [15] proposes divisive clustering and $K$-medians clustering methods to create clustering on random data. Lloyd et al. [16] proposed quantum $K$-means and nearest centroid algorithms using an efficient subroutine for quantum distance estimation, assuming, as we do, quantum access to the data. Given a dataset of $N$ vectors in a feature space of dimension $d$, the running time of each iteration of the clustering algorithm (using a distance estimation procedure with error) is $O(kN \log d)$ to produce the quantum state corresponding to the clusters. Note that the time is linear in the number of data points, and it will be linear in the dimension of the vectors if the algorithm needs to output the classical description of the clusters. In the same work, they also proposed an adiabatic algorithm for the assignment step of the $K$-means algorithm, that can potentially provide an exponential speedup.

IV. OUR SOLUTION

Our work aims to introduce the quantum concept into the SDN domain. In this section, we will first formulate the SDN controller clustering problem, following be a detailed description of our clustering method based on quantum computing. Here precision parameters are assigned as such $\delta$ for $K$-means, error parameters $\epsilon_1$ for distance estimation, $\epsilon_2$, and $\epsilon_3$ for matrix multiplication $\epsilon_4$ for tomography.

A. Problem Formulation

Given a SDN consisting of $N$ nodes (network devices or switches) at different physical locations, represented by the set $[N] = \{n_i \mid i = 1, 2, ..., N\}$. The controllers will be chosen from $[N]$, denoted as the set $[C] = \{c_j \mid j = 1, 2, ..., K\} \subset [N]$, where $K \ll N$ is the total number of controllers. For clustering, the whole network is divided into $K$ sub-networks. Every $j^{th}$ sub-network is managed by controller $c_j$, which is chosen to reduce latency $L$ between the controller and switches in its sub-network, ultimately decreasing the overall network delay. $L$ is defined as follows [4]:

$$ L = \frac{\text{distance}(m)}{2 \times 10^8 (m/s)} $$

(12)

For the quantum perspective of the problem, we start by assigning the number of clusters (sub-networks) needed for the SDN architecture. For our case study we would be using $K$ sub-networks. Theoretically, we have a set of data points stored in the memory cells of the QRAM. These data points contains the classical information, i.e., the position of node in a $2d$-plane. To begin with the mechanics of the problem we must know, a quantum state vector $|v\rangle$ for $v \in \mathbb{R}^d$ is defined as

$$ |v\rangle = \frac{1}{||v||} \sum_{m \in [d]} v_m |m\rangle, $$

(13)

where $|m\rangle$ represents $e_m$, the $m^{th}$ vector in the standard basis. The dataset is represented by a matrix $V \in \mathbb{R}^{N \times d}$, i.e., each row is a vector $v_i \in \mathbb{R}^d$ for $i \in [N]$, that represents a singular data point. The cluster centers, called centroid, at iteration $t$ are stored in the matrix $C^t \in \mathbb{R}^{K \times d}$, such that the $j^{th}$ row $c^t_j$ for $j \in \{1, 2, ..., K\}$ represents the centroid of the cluster $C^t_j$. Here we would see, that QML algorithm would work as a black box which produces $K$ centroids with lesser number of queries as compared to classical one.

B. Quantum SDN Controller Clustering

To solve the formulated problem in the previous section, our method applies quantum-based $q$-means algorithm in four steps, which will be introduced below.
1) Centroid Distance Estimation: The first step of the algorithm estimates the square distance between data points and cluster it, using a quantum procedure. The distance estimation theorem needed for the $q$-means algorithm [6] is as follows: Let a data matrix $V \in \mathbb{R}^{N \times d}$, and a centroid matrix $C \in \mathbb{R}^{K \times d}$ be stored in QRAM, such that the following unitaries $|i⟩|0⟩ \rightarrow |i⟩|v_i⟩$, and $|j⟩|0⟩ \rightarrow |j⟩|c_j⟩$ can be performed in times $O(\log(Nd))$ when the norms of the vectors are known. Then, for any $\Delta > 0$, and $\epsilon > 0$, there exists a quantum algorithm that performs the mapping [16]

$$
\frac{1}{\sqrt{N}} \sum_{i=1}^{N} |i⟩ \otimes_{j \in \{1,2,\ldots,K\}} (|j⟩|0⟩)
\rightarrow \frac{1}{\sqrt{N}} \sum_{i=1}^{N} |i⟩ \otimes_{j \in \{1,2,\ldots,K\}} (|j⟩|d^2(v_i, c_j)⟩),
$$

where $d^2(v_i, c_j) = d^2(v_i, c_j) \leq \epsilon_4$ with probability at least $1 - 2\Delta$, in time $O(k \frac{N}{\epsilon_4} \log(1/\Delta))$ where $\mu = \max_i(||v_i||^2)$.

2) Cluster assignment: At the end of step 1, we have coherently estimated the square distance between each point in the dataset and the $K$ centroids in separate registers. The $2^{nd}$ step is to select the index $j$, which corresponds to assigning each of the vectors of the dataset a particular cluster. This assignment depends on the minimum distance between a particular $v_i$ to each of the $c_j$ written as $l(v_i) = \arg\min_{j \in \{1,2,\ldots,K\}} d(v_i, c_j)$.

The circuit for finding the minimum is as follows [6]: Given $K$ different log-$p$-bit registers $\otimes_{j \in \{1,2,\ldots,K\}} |a_j⟩$, there is a quantum circuit $U_{\text{min}}$ that maps $\otimes_{j \in \{1,2,\ldots,K\}} |a_j⟩$ to $\arg\min_j (a_j)$ in time $O(K \log p)$.

In Step 2, the cost of finding the minimum is $O(K)$ for $q$-means algorithm, while we also need to undo the computation by repeating Step 1. After applying the circuit for finding the minimum and undoing the computation, the state obtained is

$$
|\psi^t⟩ := \frac{1}{\sqrt{N}} \sum_{i=1}^{N} |i⟩|l^t(v_i)⟩.
$$

3) Centroid State Creation: The previous step gave us the state $|\psi^t⟩$. The first register of this state stores the position of the node, while the second register stores the controller label for the node in the respective iteration. Given these states, we need to find the new centroids $|c_j^{t+1}⟩$, which are the average of the data points having the same label. Let $X_j^t \in \mathbb{R}^N$ be the characteristic vector for cluster $j \in \{1,2,\ldots,K\}$ at iteration $t$ scaled to unit $l_1$ norm, that is $(X_j^t)_i = \frac{1}{|C_j|} \sum_{i \in C_j} |v_i⟩$ if $i \in C_j$, and 0 if $i \notin C_j$. In fact, $|\psi^t⟩$ can be written as a weighted superposition of the characteristic vectors of the clusters.

$$
|\psi^t⟩ = \sum_{j=1}^{k} \sqrt{|C_j|} \left( \frac{1}{\sqrt{|C_j|}} \sum_{i \in C_j} |i⟩ \right) = \sum_{j=1}^{k} \sqrt{|C_j|} |X_j^t⟩|j⟩
$$

4) Step 4: Centroid update: In Step 4, we need to go from quantum states corresponding to the centroids to a classical description of the centroids to perform the update step. For this, we will apply the vector state tomography algorithm to the states $c_j^{t+1}$ that were created in Step 3. The vector state tomography gives us a classical estimate of the unit norm centroids within error $\epsilon_4$, that is $||c_j - \tilde{c}_j|| < \epsilon_4$. Using the approximation of the norms $||C_j||$ with relative error $\epsilon_3$ from Step 3, we can combine these estimates to recover the centroids as vectors: Let $\epsilon_4$ be the error we commit in estimating $c_j$ such that $||c_j - \tilde{c}_j|| < \epsilon_4$, and $\epsilon_3$ the error we commit in estimating the norms $||C_j||$. Then $||c_j - \tilde{c}_j|| \leq \sqrt{\mu}(\epsilon_3 + \epsilon_4) = \epsilon_{\text{centroid}}$.

V. Evaluation

We evaluate the quantum clustering algorithm by applying it on two different types of datasets and by comparison with the classical clustering algorithm, i.e., the $K$-means algorithm. We pick $K = 3$ initial centers within an area of 100km². From the network perspective, points within this area can be interpreted as geographical locations, where each data point represents the position of a particular node. For each dataset we use $N = 50$ points in total, these are divided up into three subsets of approximately equal sizes. Around the centers within a radius of $r = 12$ km we distribute the subsets

- uniformly in the first dataset and
- gaussian in the second dataset, where the radius $r$ is used as the standard deviation of the distribution.

Applying the algorithms on these datasets results in centroids which are close to the initial centers. Fig. 2 and Fig. 3 show the results for such datasets. We see that the quantum algorithm are prone to errors, particularly in assigning points to a cluster which actually belong to another one.

![Fig. 2. Cluster assignment of uniform dataset in classical machine learning algorithm (a) & quantum machine learning algorithm (b).](image)
the quantum algorithm results in slightly larger maximum latencies.

VI. SUMMARY & FUTURE WORK

This work showed that the quantum algorithm is prone to errors because of the different precision parameters. However, the idea behind using $q$-means is due to its query complexity, which provides a polylogarithmic ($O(K)$) speedup for the number of data points, compared to $K$-means’ linear time complexity. We expect our work would provide a substantial gain in overcoming the different network-related problems by using quantum science in future communication networks.

Since aim of this work was to introduce quantum concepts and possible benefits of using it on SDN systems, we assumed and simulated a specific scenario of a large-scale SDN which is in its initial design phase. With this, we determined the optimal location to place controllers in order to reduce controller-switch communication delay. As a future evaluation, we can work on already established SDN architectures and minimize different performance matrices in terms of latency, reliability, resilience, energy conservation and load balancing.

ACKNOWLEDGMENT

This work has been partially funded by the German Research Foundation (DFG, Deutsche Forschungsgemeinschaft) as part of Germany’s Excellence Strategy – EXC2050/1 – Project ID 390696704 – Cluster of Excellence “Centre for Tactile Internet with Human-in-the-Loop” (CeTI) of Technische Universität Dresden. The authors also acknowledge the financial support by the Federal Ministry of Education and Research of Germany in the programme of “Souverän. Digital.

TABLE I

<table>
<thead>
<tr>
<th>Method</th>
<th>Uniform</th>
<th>Gaussian</th>
</tr>
</thead>
<tbody>
<tr>
<td>$K$-means</td>
<td>(152.4± 8.7) ns</td>
<td>(115.4 ± 7.9) ns</td>
</tr>
<tr>
<td>$q$-means</td>
<td>(141.2 ± 11.6) ns</td>
<td>(126.3 ± 15) ns</td>
</tr>
</tbody>
</table>

REFERENCES


