Quantum Information Processing (2021) 20:294
https://doi.org/10.1007/s11128-021-03240-8

Balanced *k*-means clustering on an adiabatic quantum computer

Davis Arthur[1] · **Prasanna Date**[2]

Received: 12 August 2020 / Accepted: 24 August 2021 / Published online: 4 September 2021

Abstract
Adiabatic quantum computers are a promising platform for efficiently solving challenging optimization problems. Therefore, many are interested in using these computers to train computationally expensive machine learning models. We present a quantum approach to solving the balanced k-means clustering training problem on the D-Wave 2000Q adiabatic quantum computer. In order to do this, we formulate the training problem as a quadratic unconstrained binary optimization (QUBO) problem. Unlike existing classical algorithms, our QUBO formulation targets the global solution to the balanced k-means model. We test our approach on a number of small problems and observe that despite the theoretical benefits of the QUBO formulation, the clustering solution obtained by a modern quantum computer is usually inferior to the solution obtained by the best classical clustering algorithms. Nevertheless, the solutions provided by the quantum computer do exhibit some promising characteristics. We also perform a scalability study to estimate the run time of our approach on large problems using future quantum hardware. As a final proof of concept, we used the quantum approach to cluster random subsets of the Iris benchmark data set.

Keywords Quantum computing · Quantum machine learning · k-Means clustering · Balanced clustering

1 Introduction

Applications of machine learning are prevalent throughout the modern world. While their tasks vary greatly in purpose and scale, all machine learning models must be

✉ Davis Arthur
dsa0013@auburn.edu

Prasanna Date
datepa@ornl.gov

1 Auburn University, Auburn, AL, USA

2 Oak Ridge National Laboratory, Oak Ridge, TN, USA

trained before they can be deployed for practical use. In some cases, the training process is extremely time-consuming, even on the most powerful classical computers. This is particularly true for models with NP-hard or NP-complete training problems such as k-means clustering [1], neural networks [2], decision tree learning [3], etc.

Quantum computers offer an alternative platform for efficiently solving computationally challenging problems. For instance, the D-Wave 2000Q adiabatic quantum computer approximately solves the NP-complete quadratic unconstrained binary optimization (QUBO) problem efficiently. The D-Wave quantum computer has already been used for a number of machine learning tasks including training a support vector machine [4], training a restricted Boltzmann machine [5,6], linear regression [7], and matrix factorization for feature learning [8]. While modern quantum computers are too small and error-prone to effectively solve large problems, their scale and fidelity are expected to improve dramatically in time [9].

In this paper, we use the D-Wave 2000Q adiabatic quantum computer to perform a special case of k-means clustering. k-Means clustering is a popular machine learning model that partitions a set of N data points into k clusters such that each cluster is made up of similar points. Similarity is measured by the statistical variance within each cluster. We will focus on balanced k-means clustering, which requires that each cluster contains approximately the same number of points. Balanced clustering models are used in a variety of domains including network design [10], marketing [11], and document clustering [12].

Classically, it is computationally challenging to find the exact solution to the balanced k-means training problem. Thus, existing algorithms converge after finding a locally optimal solution. In the worst case, this can still require large computational resources, especially as problem size scales. Due to these challenges, we explore the prospect of training the balanced k-means model on an adiabatic quantum computer. First, we outline a QUBO formulation of the balanced k-means clustering training problem. We then theoretically analyze our formulation, comparing our quantum approach to current classical algorithms. Next, we empirically analyze the clustering performance and scalability of our quantum approach on synthetic classification data sets. Finally, we analyze the clustering performance of our approach on portions of the Iris benchmark data set.

2 Related work

The k-means clustering model is one of the most widely used unsupervised machine learning techniques. Classically, the model is usually trained through an iterative approach known as Lloyd's algorithm. Hartigan and Wong show that the time complexity of this approach is $\mathcal{O}(Nkdi)$ where N is the number of data points, k is the number of clusters, d is the dimension of the data set, and i is the number of iterations before the algorithm converges [13]. Arthur and Vassilvitskii prove that for random cluster initialization, $i = 2^{\Omega(\sqrt{n})}$ with high probability [14]. Therefore, Lloyd's algorithm has superpolynomial time complexity.

Many different implementations and variations of Lloyd's algorithm have been proposed to avoid long training times or poor clustering performance. Na et al. propose an efficient implementation that reduces the number of required distance calculations without compromising clustering quality [15]. Celebi et al. compare the impact of several different centroid initialization methods on clustering performance and run time [16]. Kapoor and Singhal observe a reduction in run time and superior clustering results when sorting input data before training the k-means model [17]. The Scikit-learn implementation of Lloyd's algorithm bounds the number of iterations by a constant, effectively reducing the time complexity to $\mathcal{O}(Nkd)$ [18]. We have used this implementation as a point of comparison to our quantum approach.

Constrained k-means models, such as balanced k-means clustering, are common in applications where additional knowledge regarding the training data or the form of a plausible solution is known. Sometimes constrained k-means models are also used in instances where the generic k-means algorithm is likely to converge to a suboptimal solution [19]. Bennett et al. propose an algorithm that enforces a minimum bound on cluster size [19]. This approach reduces to balanced clustering when the minimum cluster size is $\lfloor N/k \rfloor$. Ganganath et al. present a constrained k-means clustering algorithm in which the size of each cluster is specified prior to training the model [20]. Malinen et al. propose an efficient balanced k-means clustering algorithm that runs in $\mathcal{O}(N^3)$ time [21]. This algorithm will be used as a point of comparison to our quantum approach.

Quantum approaches to training clustering models have been proposed as well. Khan et al. implement a quantum algorithm similar to Lloyd's algorithm on the IBMQX2 universal quantum computer [22]. Ushijima-Mwesigwa et al. demonstrate partitioning a graph into k parts concurrently using quantum annealing on the D-Wave 2X machine [23]. Neukart et al. propose a quantum-classical hybrid approach to clustering [24]. Wereszczynski et al. demonstrate the performance of a novel quantum clustering algorithm on small data sets using the D-Wave 2000Q [25]. Bauckhage et al. propose QUBO formulations for binary clustering ($k = 2$) [26] and k-medoids clustering [27]. Kumar et al. present a QUBO formulation for k-clustering that approximates the k-means model [28]. We have previously formulated three machine learning problems as QUBO problems [29].

While many quantum clustering algorithms have been proposed, none target the exact solution to the k-means or balanced k-means clustering model. Instead, they are heuristic approaches that approximate the k-means optimization problem. We propose a QUBO formulation that is identical to the balanced k-means training problem. We also tested our approach on synthetic and benchmark data sets.

3 QUBO formulation

Adiabatic quantum computers are able to find the global minimum of the quadratic unconstrained binary optimization (QUBO) problem, which can be stated as follows:

$$\min_{z \in \mathbb{B}^M} z^T A z \tag{1}$$

where $\mathbb{B} = \{0, 1\}$ is the set of binary numbers, $z \in \mathbb{B}^M$ is the binary decision vector, and $A \in \mathbb{R}^{M \times M}$ is the real-valued $M \times M$ QUBO matrix. Our goal is to convert the balanced k-means training problem into this form.

The k-means clustering model, aims to partition a data set $X = \{x_1, x_2, \ldots, x_N\}$ into k clusters $\Phi = \{\phi_1, \phi_2, \ldots, \phi_k\}$. The centroid of cluster ϕ_i is denoted as μ_i. Formally, training the k-means clustering model is expressed as:

$$\min_{\Phi} \sum_{i=1}^{k} \sum_{x \in \phi_i} ||x - \mu_i||^2 \tag{2}$$

Utilizing the law of total variance, the training problem can be rewritten as:

$$\min_{\Phi} \sum_{i=1}^{k} \frac{1}{2|\phi_i|} \sum_{x,y \in \phi_i} ||x - y||^2 \tag{3}$$

In the case that each cluster is of equal size (i.e., balanced), $|\phi_i|$ is constant, and Problem 3 reduces to:

$$\min_{\Phi} \sum_{i=1}^{k} \sum_{x,y \in \phi_i} ||x - y||^2 \tag{4}$$

To formulate Problem 4 as a QUBO problem, it will be useful to define a matrix $D \in \mathbb{R}^{N \times N}$ where each element is given by $d_{ij} = ||x_i - x_j||^2$. We also define a binary matrix $\hat{W} \in \mathbb{B}^{N \times k}$ such that $\hat{w}_{ij} = 1$ if and only if point x_i belongs to cluster ϕ_j. This use of binary variables is identical to the “one-hot encoding" quantum clustering method proposed by Kumar et al. [28]. Since we are assuming clusters of the same size, each column in $\hat{W}$ should have approximately N/k entries equal to 1. Additionally, since each data point belongs to exactly one cluster, each row in $\hat{W}$ must contain exactly one entry equal to 1. Using this notation, the inner sum in Problem 4 can be rewritten:

$$\sum_{x,y \in \phi_j} ||x - y||^2 = \hat{w}'^T_j D \hat{w}'_j \tag{5}$$

where $\hat{w}'_j$ is the jth column in $\hat{W}$. From this relation, we can cast Problem 4 into a constrained binary optimization problem. First, we vertically stack the Nk binary variables in $\hat{W}$ as follows:

$$\hat{w} = [\hat{w}_{11} \ldots \hat{w}_{N1}\ \hat{w}_{12} \ldots \hat{w}_{N2} \ldots \hat{w}_{1k} \ldots \hat{w}_{Nk}]^{\mathrm{T}} \tag{6}$$

Provided the constraints on $\hat{w}$ are upheld, Problem 4 is equivalent to:

$$\min_{\hat{w}} \hat{w}^T (I_k \otimes D) \hat{w} \tag{7}$$

where I_k is the k-dimensional identity matrix.

We can remove the constraints on $\hat{w}$ by including penalty terms that are minimized when all conditions are satisfied. First, we account for the constraint that each cluster must contain approximately N/k points. For a given column $\hat{w}'_j$ in $\hat{W}$, this can be enforced by including a penalty of the form:

$$\alpha(\hat{w}'^T_j \hat{w}'_j - N/k)^2 \tag{8}$$

where α is a constant factor intended to make the penalty large enough that the constraint is always upheld. Dropping the constant term $\alpha(N/k)^2$, this penalty is equivalent to $\hat{w}'^T_j \alpha F \hat{w}'_j$ where F is defined as:

$$F = 1_N - \frac{2N}{k} I_N \tag{9}$$

In the expression above, 1_N refers to an $N \times N$ matrix where each element is equal to 1. Using this formulation, the sum of all column constraint penalties is:

$$\hat{w}^T (I_k \otimes \alpha F)\hat{w} \tag{10}$$

Next, we account for the constraint that each point belongs to exactly 1 cluster. For a given row $\hat{w}_i$, this can be enforced by including a penalty of the form:

$$\beta(\hat{w}_i^T \hat{w}_i - 1)^2 \tag{11}$$

where β is a constant with the same purpose as α in Eq. 8. Dropping the constant term, this penalty is equivalent to $\hat{w}_i^T \beta G \hat{w}_i$ where G is defined as:

$$G = 1_k - 2I_k \tag{12}$$

To find the sum of all row constraint penalties, we first convert the binary vector $\hat{w}$ into the form $\hat{v}$ shown below:

$$\hat{v} = [\hat{w}_{11} \ldots \hat{w}_{1k}\ \hat{w}_{21} \ldots \hat{w}_{2k} \ldots \hat{w}_{N1} \ldots \hat{w}_{Nk}]^{\mathrm{T}} \tag{13}$$

This can be accomplished through a linear transformation $Q\hat{w}$ where each element in $Q \in \mathbb{B}^{Nk \times Nk}$ is defined as:

$$q_{ij} = \begin{cases} 1 & j = N \bmod(i-1, k) + \lfloor \frac{i-1}{k} \rfloor + 1 \\ 0 & \text{else} \end{cases} \tag{14}$$

After the transformation, the sum of all row constraint penalties is given by $\hat{v}^T (I_N \otimes \beta G)\hat{v}$. This sum can be equivalently expressed as:

$$\hat{w}^T Q^T (I_N \otimes \beta G) Q \hat{w} \tag{15}$$

Combining the column and row penalties with the constrained binary optimization problem from Eq. 7, Problem 4 can be rewritten as:

$$\min_{\hat{w}} \hat{w}^T (I_k \otimes (D + \alpha F) + Q^T (I_N \otimes \beta G) Q) \hat{w} \tag{16}$$

This is identical to Eq. 1 with $z = \hat{w}$ and $A = (I_k \otimes (D + \alpha F) + Q^T (I_N \otimes \beta G) Q)$. Thus, we have converted the balanced k-means training problem (Eq. 4) into a QUBO problem which can be solved on adiabatic quantum computers. Provided N is divisible by k, and α and β are large enough to ensure all constraints are upheld, Problem 16 and Problem 4 share the same global solution.

3.1 Implementation details

In order to achieve good performance on quantum hardware, α and β must be chosen such that the penalty for violating a constraint is large, but not so large as to overshadow the importance of minimizing within cluster variance. Therefore, it is important to consider how α and β affect the range of entries in either penalty matrix. All coefficients in the column penalty matrix, F, lie within the range $[-\alpha/\alpha_0, \alpha]$, where α_0 is defined below:

$$\alpha_0 = \frac{1}{2N/k - 1} \tag{17}$$

All coefficients in the row penalty matrix, G, lie within the range $[-\beta, \beta]$.

Since each problem is eventually scaled to fit within the coupler range of the D-Wave solver, α/α_0 and β should not be significantly larger than the maximum coefficient in the distance matrix, D. Otherwise the variation between coefficients in D may fall below the precision of the D-Wave solver after scaling. In our implementation, we first scale D by a factor of $1/d_{\max}$ to ensure its largest coefficient is equal to 1. Using this convention, we expect more consistent and accurate results when α/α_0 and β are not substantially larger than 1. In Sect. 4.2.5, we perform an empirical study to estimate the optimal values of α and β for several specific problem sizes.

In practice, the quantum annealing process is not perfect, and instances occur in which a point is assigned to multiple clusters or not assigned to any cluster at all. We propose two post-processing algorithms to handle such violations. The first post-processing algorithm, which we refer to as “strict" balanced clustering, guarantees that each of the k clusters contains exactly N/k points. The second post-processing algorithm, which we refer to as “relaxed" balanced clustering, does not place any constraints on cluster size. Despite this, we expect the “relaxed" post-processing algorithm to return approximately equal size clusters since solutions of this form are energetically favorable in the QUBO formulation.

In both post-processing algorithms, we begin by scanning through the solution vector, $\hat{w}$, to determine which points were assigned to exactly one cluster. We then add each of these points to its assigned cluster one by one. In the “strict" post-processing algorithm, before we add a point to its assigned cluster, we check that the assigned cluster does not already contain N/k points. If the assigned cluster does already contain N/k points, the point is not yet assigned. Both algorithms then estimate the centroid of each cluster based only on the points that have already been assigned. If a cluster has not been assigned any points, its estimated centroid is a d-dimensional zero vector.

Next, we scan through $\hat{w}$ a second time and assign each point that has not yet been assigned. In the “relaxed" post-processing algorithm, each unassigned point is assigned to the nearest cluster. In the “strict" post-processing algorithm, each unassigned point is assigned to the nearest cluster that does not already contain N/k points. For both algorithms, each time a point is assigned to a cluster, that cluster’s centroid is updated. Once all points have been assigned to a cluster, we return the cluster assignments.

4 Results and analysis

4.1 Theoretical analysis

The generic k-means clustering problem stated in Eq. 2 and the balanced k-means clustering problem stated in Eq. 4 both contain $\mathcal{O}(Nd)$ data and $\mathcal{O}(N)$ variables (where each variable indicates the cluster assignment of a given data point). In our QUBO formulation of balanced k-means clustering, we introduce k binary variables for each variable in the original problem. Thus, the total number of variables in Eq. 16 is $\mathcal{O}(Nk)$. This translates to a quadratic qubit footprint of $\mathcal{O}(N^2k^2)$ using an efficient embedding algorithm such as [30].

It has been shown to require $\mathcal{O}(N^{kd+1})$ time to exactly solve the generic k-means clustering problem (Problem 2) [31]. Alternatively, a locally optimal solution can be found in $\mathcal{O}(Nkdi)$ time using Lloyd’s algorithm. The Scikit-learn approach to k-means is able to effectively reduce the time complexity to $\mathcal{O}(Nkd)$ by bounding the number of iterations by a constant and performing Lloyd’s algorithm multiple times from different centroid initializations. While this approach cannot guarantee a locally optimal solution, it achieves high quality clustering performance in practice.

The time complexity required to exactly solve the balanced k-means clustering problem has not been thoroughly analyzed. However, a locally optimal solution to Problem 4 can be found in $\mathcal{O}(N^3)$ time using the classical approach proposed by Malinen et al. [21]. To compare this to our quantum approach, we first determine the time complexity for converting Eq. 4 into a QUBO problem. To do so, we rewrite Eq. 16 as follows:

$$\min_{W} \sum_{l=1}^{k}\sum_{j=1}^{N}\sum_{i=1}^{N}\sum_{m=1}^{d} w_{il}(x_{im}-x_{jm})^2 w_{jl} + \alpha \sum_{l=1}^{k}\sum_{j=1}^{N}\sum_{i=1}^{N} w_{il} f_{ij} w_{jl} + \beta \sum_{l=1}^{N}\sum_{j=1}^{k}\sum_{i=1}^{k} w_{li} g_{ij} w_{lj} \tag{18}$$

From Eq. 18, the worst case time complexity in forming the QUBO matrix is $\mathcal{O}(N^2kd)$, which is dominated by the first term. In our actual implementation, we also determine the maximum element in $D \in \mathbb{R}^{N \times N}$ and scale D accordingly. This can be done in $\mathcal{O}(N^2)$ time, so it does not alter the worst case time complexity of forming the QUBO matrix. For practical purposes, solving the QUBO problem through quantum annealing can be done in constant time.

Next, we post-process the binary solution to determine our final cluster assignments. In the first step of both proposed post-processing algorithms, we scan through all Nk elements of $\hat{w}$. This requires $\mathcal{O}(Nk)$ time. Next, we compute each cluster centroid estimate using only the points that were assigned to exactly one cluster. This requires $\mathcal{O}(Nd)$ time. In the final step of both post-processing algorithms, we calculate the distance of each unassigned point to each estimated cluster centroid. In the worst case, this requires Nk distance calculations, each of which require $\mathcal{O}(d)$ time. Therefore, the worst case time complexity of both post-processing approaches is $\mathcal{O}(Nkd)$.

Even after post-processing is considered, the time complexity of the entire quantum clustering approach is dominated by the time required to formulate the QUBO matrix. Therefore, the time complexity of the quantum approach is $\mathcal{O}(N^2kd)$. Provided $kd < N$, this time complexity is better than the time complexity of the best classical balanced k-means clustering algorithm that can guarantee a local solution to the clustering problem ($\mathcal{O}(N^3)$). However, it is worse than the Scikit-learn implementation of generic k-means clustering ($\mathcal{O}(Nkd)$).

4.2 Empirical analysis

4.2.1 Methodology and performance metrics

Our QUBO approach was tested using the D-Wave 2000Q adiabatic quantum computer as well as D-Wave's simulated annealing sampler. We compare the performance of our approach to the Scikit-learn implementation of classical k-means as well as our own implementation of the classical balanced k-means algorithm with the best time complexity [21]. Note that the Scikit-learn implementation of k-means searches for a solution to Problem 2, while the classical balanced k-means algorithm and our QUBO approach search for a solution to Problem 4. The Scikit-learn algorithm is still a valid point of comparison since the solution to both problems should be very similar for all data sets used in our experiments. We also determined the exact solution to each balanced clustering problem using the Gurobi Optimizer mixed integer programming solver.

We use two performance metrics to compare the clustering algorithms: (i) inertia (see Sect. 4.2.2) and (ii) total computing time. In the quantum approach, total computing time is composed of the time required to convert the problem into a QUBO problem, the time required to embed the QUBO problem on the hardware, the time for the quantum computer to solve the QUBO problem (total sampling time), and the time required to extract the clustering information from the binary solution (post-processing time).

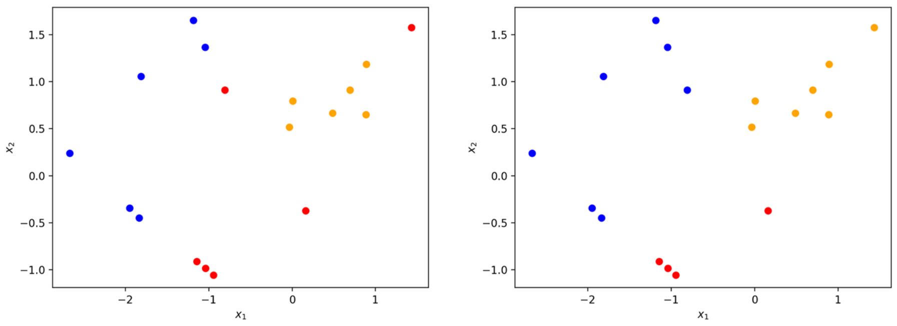

Fig. 1 Example of a synthetic data set clustered by the QUBO approach with quantum annealing using "strict" post-processing (left) and "relaxed" post-processing (right). The data set contains $N = 18$ points and $k = 3$ classes

4.2.2 Inertia

We used the value of the objective function of the generic k-means model as our primary metric to quantify clustering performance. This quantity is commonly referred to as inertia. The ideal balanced clustering of a data set will minimize inertia while maintaining equal size clusters. Using the notation defined in Sect. 3, inertia is defined as follows:

$$\text{Inertia} = \sum_{i=1}^{k} \sum_{x \in \phi_i} ||x - \mu_i||^2 \tag{19}$$

4.2.3 Data generation

We tested our algorithm on synthetic classification data sets created using the *make_classification* function in the Scikit-learn data sets package. Each data set contains N points, k classes, 1 cluster per class, and d features. This function generates a data set where each cluster is centered at one of the vertices of a d-dimensional hypercube with side length 2.0. The points are then generated from a normal distribution (standard deviation of 1.0) about their cluster center. For all experiments, each class was made up of exactly N/k points (Fig. 1).

4.2.4 Hardware configuration

Pre-processing and post-processing for our quantum approach and entire classical approach were run on a machine with 2.7 GHz Dual-Core Intel i5 processor and 8 GB 1,867 MHz DDR3 memory. The quantum approach also used the D-Wave 2000Q quantum computer, which had 2048 qubits and about 5600 inter-qubit connections. All problems discussed in Sects. 4.2.6 and 4.3 were mapped to the quantum hardware using the *EmbeddingComposite* class from the D-Wave library. By default, this class scales each problem to use the full coupler range of the hardware, and then embeds each problem using the *minorminer.find_embedding* function. Broken chains were handled using majority vote. No additional D-Wave post-processing (such as optimization post-processing or sampling post-processing) was used. For all experiments, each quantum annealing operation is performed 200 times, and only the lowest energy sample is used.

4.2.5 Optimal values of α and β

As previously mentioned, the performance of the QUBO model is largely dependent on the values chosen for α and β. To estimate the optimal values of these factors, we clustered 9 synthetic data sets on the quantum computer using 64 different (α, β) combinations. The “strict" post-processing method was used for all problems. Each of the 9 data sets used to test the quantum approach represented a unique problem type, defined by the number of points and number of clusters in the data set. To ensure each data set accurately represented its problem type, we checked that the exact balanced clustering solution of each data set had an inertia value similar to other data sets of the same problem type.

In the 64 trials for each problem, we varied α linearly from 0 to $8\alpha_0$, and we varied β linearly from 0 to 2. In preliminary tests, we found that the quantum approach usually performed poorly when α or β were outside of these ranges. In Fig. 2, we plotted an 8×8 pixel image for each tested data set. Each pixel is colored based on the inertia of the quantum clustering approach when a particular (α, β) pair was used. The value of each pixel, denoted as z, is defined below:

$$z(\alpha, \beta) = \frac{(\text{Exact Solution Inertia}) - (\text{Quantum Balanced Inertia})_{\alpha\beta}}{(\text{Exact Solution Inertia})} \tag{20}$$

In each plot, a pixel corresponding to an (α, β) pair that performed relatively well is colored dark blue, while a pixel corresponding to an (α, β) pair that performed relatively poorly is colored white. A z value of zero indicates the quantum approach found the exact solution. For all problems except (15, 3) and (21, 3), the quantum approach found the exact solution for at least one (α, β) pair.

We estimated optimal values for α and β for each problem type by inspecting the plots in Fig. 2. An ideal (α, β) pair should be located in an area that is concentrated in dark blue. The (α, β) pairs we chose for each problem type are reported in Table 1. This table also includes three problems types ((8, 2), (9, 3), and (18, 3)) that we did not test over 64 different (α, β) pairs. For these problems, we chose α and β based on the values we had chosen for other problems with the same number of clusters and a similar number of points.

4.2.6 Clustering synthetic data sets

We compare the clustering quality produced by quantum balanced k-means, classical balanced k-means, and classical k-means on several small synthetic data sets. The synthetic data sets can be categorized by the number of points and the number of classes (N, k). For each category, all clustering approaches were run on 10 unique data sets. For each trial, the QUBO approach is performed once using the D-Wave

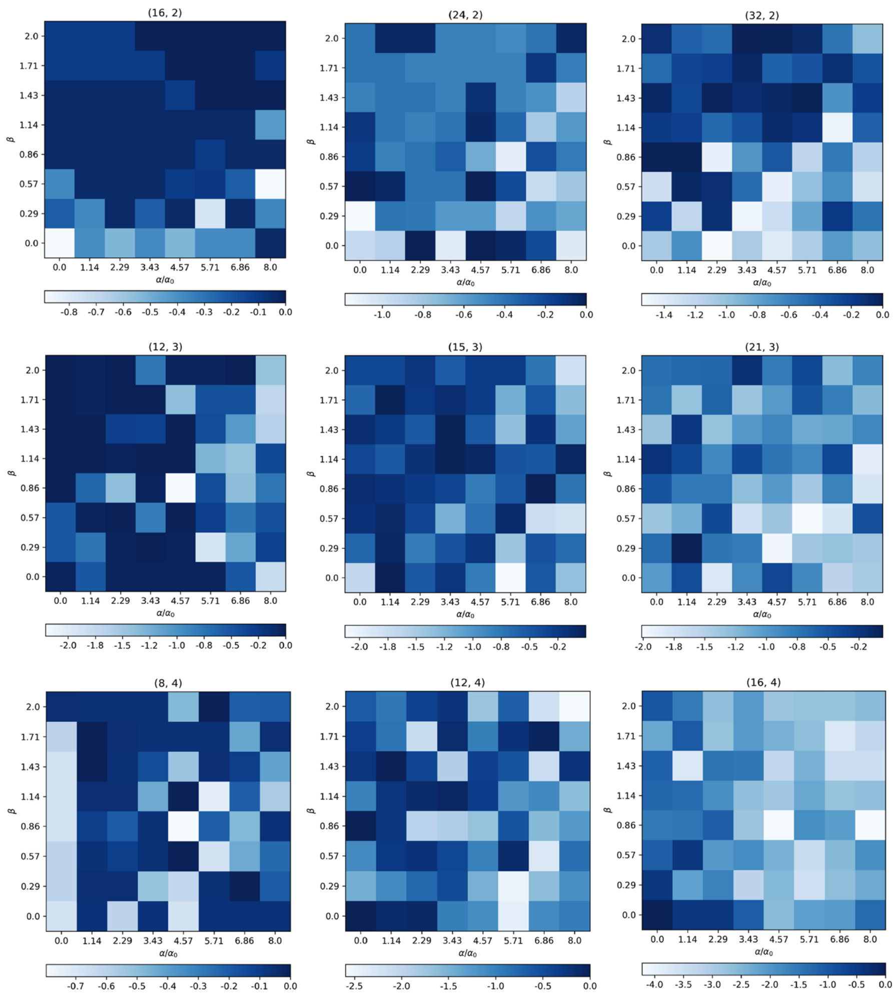

Fig. 2 Relative inertia (see Eq. 20) of the quantum approach with “strict" post-processing on 9 synthetic data sets. The labels indicate the number of points and number of clusters (N, k) in each data set. For each problem, 64 different (α, β) combinations were tested, and α_0 was calculated using Eq. 17

Table 1 Estimated optimal constraint penalty factors for each problem type based on the results shown in Fig. 2

(N, k)	(α, β)	(N, k)	(α, β)	(N, k)	(α, β)
(8, 2)	(0.5, 1.45)	(9, 3)	(0.5, 1.15)	(8, 4)	(2.1, 0.25)
(16, 2)	(0.4, 1.45)	(12, 3)	(0.5, 1.15)	(12, 4)	(0.45, 1.15)
(24, 2)	(0.2, 1.15)	(15, 3)	(0.45, 1.15)	(16, 4)	(0.15, 0.85)
(32, 2)	(0.15, 1.45)	(18, 3)	(0.45, 1.15)		
		(21, 3)	(0.1, 1.15)		

These α and β factors were used in each of the clustering experiments discussed in Sects. 4.2.6 and 4.3

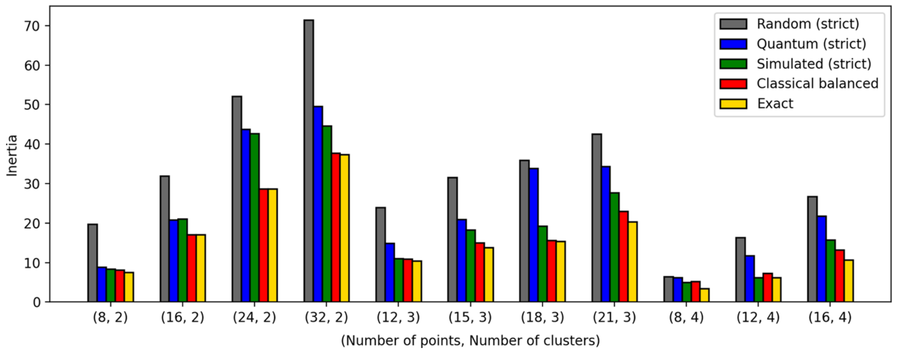

Fig. 3 Average inertia of all clustering approaches that guarantee equal size clusters. The average inertia of random binary solutions are included for comparison. Exact data is reported in Table 2

2000Q machine and once using D-Wave's simulated annealing sampler. The returned solutions are then analyzed once using the "strict" post-processing algorithm and once using the "relaxed" post-processing algorithm. We also performed both post-processing algorithms on a pseudo-solution to the QUBO problem, consisting of Nk random bits. As a final point of comparison, we determined the exact solution to each balanced clustering problem using the Gurobi Optimizer mixed integer programming solver. This exact solution uses the constrained formulation of the balanced clustering problem ensuring that each cluster contains exactly N/k points. The average inertia value of each clustering approach is reported in Table 2.

Note that for both quantum and simulated annealing, the same α and β factors are used for each problem type. These α and β factors are reported in Table 1. Also note that the simulated annealing results are generated using D-Wave's default parameters. These parameters call for the simulated annealing algorithm to be performed exactly once (per experiment) on a randomly generated initial state. The simulated annealing algorithm performs 1000 sweeps using a geometric annealing schedule.

First, consider the performance of the clustering approaches that guarantee equal size clusters (see Fig. 3). For each problem type except (8, 4) and (12, 4), the QUBO approach performed worse than the classical balanced algorithm. This may be in part because α and β are not large enough to ensure the global solution to our QUBO model is the exact solution to the balanced clustering problem. Unfortunately, when larger penalty factors were used, the importance of minimizing within cluster variance was significantly compromised. Despite its shortcomings, the QUBO approach significantly outperforms the randomly generated solution for almost all problem types.

Simulated annealing produced a better solution than quantum annealing on average. For some problem types, such as (18, 3) and (12, 4), the discrepancy is particularly apparent. We suspect the disparity in performance between simulated annealing and quantum annealing has two causes. First, the quantum hardware is imperfect. Broken chains occur during the annealing process, and the computer does not always find the most energetically favorable solution to the QUBO problem. Second, when the problem is scaled to fit the quantum hardware, the variation between coefficients of the QUBO matrix may fall below the precision of the D-Wave machine.

Table 2 Average inertia values of the QUBO approach, classical balanced k-means, and classical k-means (Scikit-learn) when clustering synthetic data sets

(N, k)	Quantum annealing (strict)	Simulated annealing (strict)	Classical balanced	Exact	Quantum annealing (relaxed)	Simulated annealing (relaxed)	Scikit-learn
(8, 2)	8.88	8.44	8.19	7.54	7.41	7.35	6.11
(16, 2)	20.85	21.00	17.04	17.04	16.23	16.65	15.05
(24, 2)	43.75	42.67	28.70	28.64	29.23	27.22	23.50
(32, 2)	49.60	44.61	37.68	37.33	40.34	37.00	35.31
(12, 3)	14.87	11.03	10.88	10.43	10.14	9.71	7.88
(15, 3)	20.88	18.24	14.99	13.84	16.60	13.07	10.94
(18, 3)	33.92	19.28	15.63	15.38	27.30	14.03	12.28
(21, 3)	34.29	27.68	23.00	20.31	22.75	18.43	16.93
(8, 4)	6.17	4.97	5.24	3.38	4.19	3.47	1.87
(12, 4)	11.80	6.21	7.25	6.17	10.48	6.21	4.85
(16, 4)	21.72	15.71	13.26	10.67	16.29	9.46	7.80

The QUBO approach is performed once with simulated annealing and once with quantum annealing. In each case, both post-processing algorithms are performed. The average inertia of the exact solution to the balanced clustering problem is included for reference

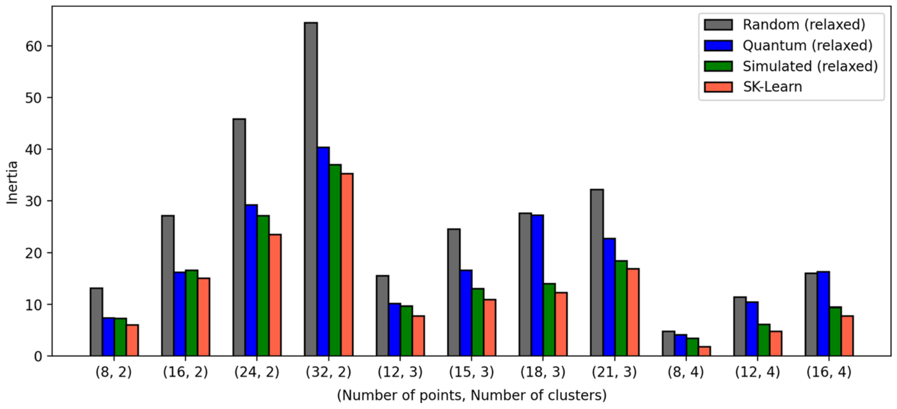

Fig. 4 Average inertia of all clustering approaches that do not guarantee equal size clusters. The average inertia of random binary solutions are included for comparison. Exact data is reported in Table 2

Next, consider the performance of the clustering approaches that do not guarantee equal size clusters (see Fig. 4). Notice that many of the inertia values fall below the inertia of the exact solution to the balanced clustering problem. This is not surprising since each of these approaches can return clusters of unequal size when favorable. For each problem type, the Scikit-learn implementation of classical k-means achieved the lowest average inertia value. The QUBO approach with simulated annealing performed slightly worst, but its inertia value was still less than the exact solution for most problem types. Again, the QUBO approach performed noticeably worse when quantum annealing was used.

It is worth considering the severity with which the equal size cluster constraint was compromised by the clustering approaches shown in Fig. 4. For each problem type, we determined the size of all clusters produced by the QUBO approach after “relaxed" post-processing. We then computed the coefficient of variation over all cluster sizes. As a point of comparison, we also computed the coefficient of variation over all cluster sizes produced by the Scikit-learn implementation of classical k-means. The coefficient of variation is defined as the standard deviation (σ) divided by the mean (μ). Note that the mean cluster size is always N/k.

$$\text{Coefficient of variation} = \frac{\sigma}{\mu} \tag{21}$$

In Fig. 5 we observe that the QUBO approach (with both simulated and quantum annealing) usually produces clusters with less variation in size than the classical k-means algorithm. This is not surprising, since balanced clusters are energetically favorable in the QUBO formulation. It should also be noted that by increasing α in the QUBO model, it is possible to more strongly enforce the constraint of equal size clusters. However, this typically comes at the cost of sub-optimal cluster assignments.

For each trial, we also determined the number of clusters assigned to each point and the number of points assigned to each cluster before post-processing. We express this data in Table 3 by providing the frequency distribution of the number of clusters

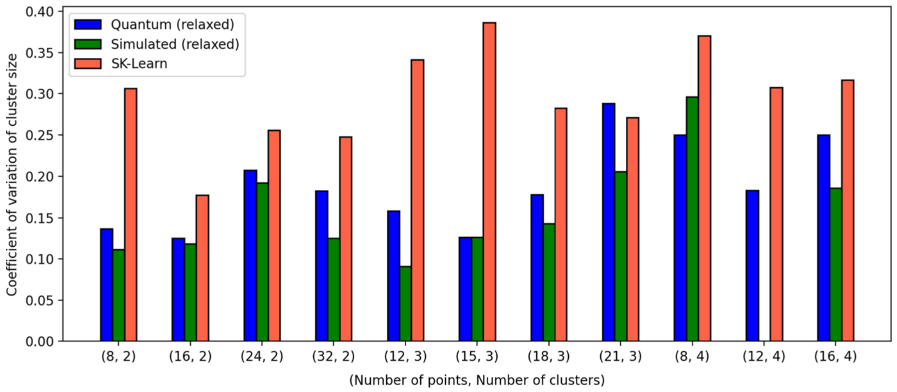

Fig. 5 Coefficient of variation of cluster sizes produced by the QUBO approach with quantum annealing and simulated annealing using the "relaxed" post-processing algorithm. The coefficient of variation of cluster sizes produced by the Scikit-learn implementation of classical *k*-means is also included.

assigned to a point (expressed as a percentage of all points) as well as the frequency distribution of the number of points assigned to a cluster (expressed as a percentage of all clusters).

For all problem types, the vast majority of points were assigned to exactly 1 cluster. When violations occurred, it was much more common for a point to be assigned to 0 clusters than 2 or more clusters. For problems with $k \leq 3$, the percentage of points assigned to exactly 1 cluster decreased as the number of points increased. This was not the case for problems with 4 clusters, likely because we used a smaller β factor for problems of type (8, 4) than we did for problems of type (12, 4) and (16, 4). The β factor is directly responsible for upholding the constraint that each point is assigned to exactly one cluster, and adjusting its value dramatically alters the distribution. By using an extremely large β value, we can ensure almost all points are assigned to one cluster even on larger problems. However, this significantly reduces the quantum computer's ability to minimize within cluster variance.

For small problem types such as (8, 2), (12, 3), and (8, 4), most clusters were assigned exactly N/k points. However, as problem size increased, the number of clusters assigned exactly N/k points decreased. For all problem types, it is much more common for a cluster to be assigned less than N/k points than more than N/k points. By increasing the value of α, we can better enforce the equal size cluster constraint, particularly for larger problems. However, this usually compromises the quantum computer's ability to minimize within cluster variance.

Even though the α and β factors we chose allow for violations in the QUBO solution, these factors are relatively optimal choices for minimizing inertia on modern quantum hardware after either of our post-processing algorithms is performed. This is because the points that the quantum computer assigns to exactly one cluster are relatively well clustered. When larger penalty factors are used, some points are forced into sub-optimal cluster assignments instead. In Fig. 6, we consider the subset of points that are assigned to exactly one cluster by the quantum computer. We compute the inertia of this subset based on the partitioning produced by quantum annealing, classical

Table 3 Frequency distributions of the number of clusters assigned to a point (expressed as a percentage of all points) and the number of points assigned to a cluster (expressed as a percentage of all clusters)

		Number of clusters assigned to each point					Number of points assigned to each cluster				
N	k	0	1	2	3	4	$n-3$	$n-2$	$n-1$	n	$n+1$
8	2	3.8	96.2	0.0	n/a	n/a	0.0	0.0	20.0	75.0	5.0
16	2	10.6	89.4	0.0	n/a	n/a	0.0	20.0	45.0	35.0	0.0
24	2	22.9	76.3	0.8	n/a	n/a	35.0	35.0	5.0	5.0	0.0
32	2	23.8	75.9	0.3	n/a	n/a	20.0	5.0	5.0	10.0	0.0
12	3	3.3	96.7	0.0	0.0	n/a	0.0	0.0	23.3	66.7	10.0
15	3	10.0	90.0	0.0	0.0	n/a	0.0	6.7	36.7	56.7	0.0
18	3	16.7	82.2	1.1	0.0	n/a	0.0	10.0	73.3	16.7	0.0
21	3	25.7	74.3	0.0	0.0	n/a	20.0	40.0	26.7	10.0	0.0
8	4	12.5	75.0	12.5	0.0	0.0	0.0	0.0	0.0	100.0	0.0
12	4	6.7	93.3	0.0	0.0	0.0	0.0	0.0	22.5	75.0	2.5
16	4	24.4	75.0	0.6	0.0	0.0	0.0	22.5	50.0	27.5	0.0

Situations that cannot occur, such as a point being assigned to more than k clusters or a cluster being assigned less than zero points, are denoted by "n/a." The expected number of points assigned to each cluster (N/k) is denoted by n

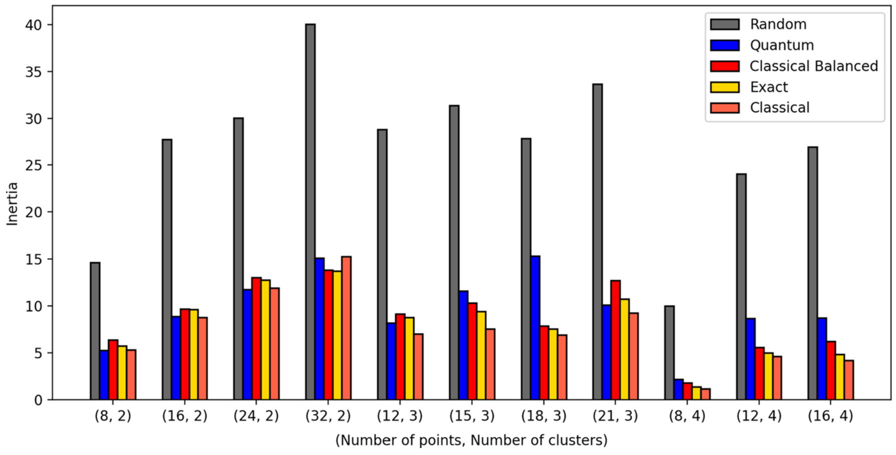

Fig. 6 Average inertia using the assignments of quantum balanced k-means, classical balanced k-means, classical k-means, and exact balanced k-means if we only consider the subset of points that were assigned to exactly one cluster by the quantum approach. For reference, the average inertia of a random clustering of these points is also included

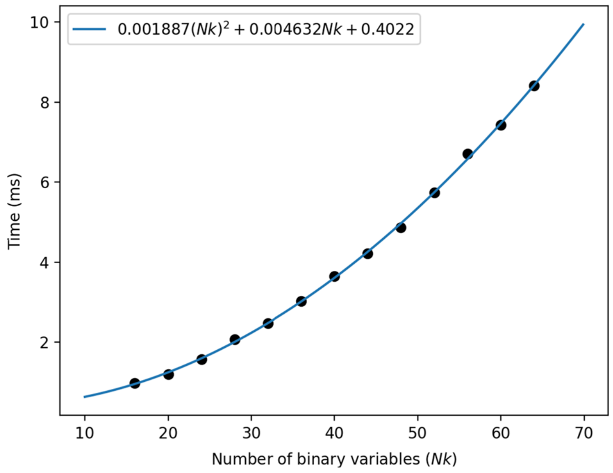

Fig. 7 Time required to embed small problems on the D-Wave using the embedding algorithm proposed by Date et. al. [30]. Embedding time scales quadratically with the number of binary variables

balanced k-means clustering, and classical k-means clustering (Scikit-learn). We also plotted the inertia of this subset if each point is assigned to one of the k clusters at random as well as the inertia if each point is assigned according to the exact balanced clustering solution of the entire data set.

For many of the problem types where $k = 2$ or $k = 3$, the clustering produced by the quantum approach has a smaller average inertia value than at least one of the classical approaches. It is worth noting that these occurrences do not indicate that the quantum approach is outperforming the other algorithms, since the goal is to minimize inertia over the entire data set, not inertia over a specific subset of points. Nevertheless, the fact the inertia values produced by the quantum computer are comparable to the inertia values produced by the assignments of the exact solution indicates that the points assigned to one cluster by the QUBO solution are well clustered. This is also illustrated by the large discrepancy between the inertia of the quantum assignments and the inertia produced by a random clustering of the subset.

4.2.7 Scalability with number of data points (*N*)

We also perform a scalability study to determine how the run time of our quantum approach varies as the number of data points increases. Due to the qubit limitations of modern adiabatic quantum computers, problems that require more than 64 binary variables ($Nk > 64$) are impossible on the D-Wave 2000Q. However, we can approximate the run time of our algorithm on larger problems by measuring the time required to formulate the QUBO problem and the time required to post-process a plausible solution. We estimate the time required to embed the problem (t_e) as well as the total sampling time (t_s).

While the D-Wave *EmbeddingComposite* class was used to embed the problems discussed in Sects. 4.2.6 and 4.3, the run time of the embedding algorithm used by this class (i.e., *minorminer.find_embedding*) can vary drastically from problem to problem, and more efficient embedding algorithms exist. In particular, the run time of the efficient embedding algorithm proposed in [30] scales quadratically with the number of binary variables in the QUBO problem. Extrapolating upon the performance of this embedding algorithm on small problems (see Fig. 7), we approximate embedding time using the following equation:

$$t_e = 1.887 \times 10^{-6}(Nk)^2 + 4.632 \times 10^{-6}(Nk) + 4.022 \times 10^{-4} \text{ s} \tag{22}$$

As previously mentioned, annealing is performed in constant time on modern quantum hardware. Therefore, we assume that the total sampling time (t_s) for larger problems is equal to the average total sampling time for the small clustering experiments discussed in Sect. 4.2.6.

$$t_s = 47.788 \text{ ms} \tag{23}$$

It should be noted that it is unknown how the run time of a future machine may behave. Improvements in the fidelity and scale of quantum hardware may require longer annealing times or a larger number of samples. Therefore, the total sampling time reported in Eq. 23 is best interpreted as a lower bound for the sampling time of a future machine.

We performed classical k-means, classical balanced k-means, and our QUBO formulation on data sets of increasing size. For each problem size, all three approaches were run on 50 synthetic classification data sets. Each data set contained $k = 4$ classes, and each data point had $d = 2$ features. The average run time of each clustering approach is reported in Fig. 8 and Table 4. For each trial, both post-processing algorithms were performed on a plausible QUBO solution. However, we have only

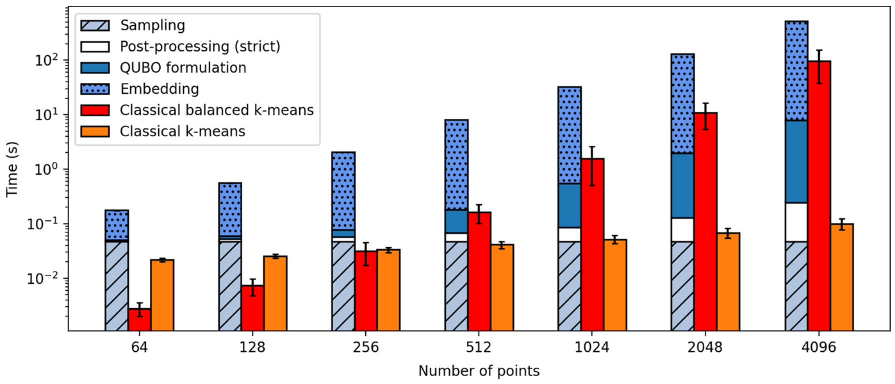

Fig. 8 Total computing time of quantum balanced k-means (blue bar), classical balanced k-means (red bar), and classical k-means (orange bar) as the number of points (N) in the training data set varies. Embedding and sampling times are approximate (Color figure online)

included the post-processing time corresponding to the "strict" post-processing method in Fig. 8 since both post-processing algorithms had extremely similar run times.

For all problem sizes, the quantum approach performed slower than both classical algorithms. However, the quantum run time was dominated by the embedding time. Embedding is extremely difficult on modern quantum computers due to limited qubit connectivity. We expect embedding will become a considerably faster process as qubit connectivity increases. Therefore, on a future quantum computer, the quantum algorithm may outperform the classical balanced k-means algorithm for $N \geq 1024$, depending on how well the embedding process is optimized. Of the three approaches, our results indicate that the Scikit-learn implementation of classical k-means scales the best. This is expected since the time complexity of the Scikit-learn implementation of classical k-means ($\mathcal{O}(Nkd)$) is better than classical balanced k-means ($\mathcal{O}(N^3)$) or quantum balanced k-means ($\mathcal{O}(N^2kd)$).

4.2.8 Scalability with number of clusters (*k*)

Following the same procedure, we analyze the scalability of each algorithm as the number of clusters k is increased. For each problem type, all three clustering algorithms were run on 50 synthetic data sets. Each data set consisted of $N = 256$ points, and all points had $d = 8$ features. The average run time of each clustering approach is reported in Fig. 9 and Table 5.

In all cases, the quantum approach had a longer run time than both classical algorithms. Additionally, the quantum run time scaled worse as the number of clusters increased. This is expected since the third term in the QUBO formulation (Eq. 18) has time complexity $\mathcal{O}(Nk^2)$. Alternatively, classical k-means scales linearly with the number of clusters ($\mathcal{O}(Nkd)$), and balanced k-means clustering scales independently of the number of clusters ($\mathcal{O}(N^3)$). It is somewhat surprising that the average run time of the balanced k-means clustering approach decreases for $k > 16$. However, we suspect this is due to the smaller cluster sizes when k is large.

Table 4 Time required to perform classical k-means, classical balanced k-means, and our QUBO formulation on data sets of increasing size

Number of points	Classical k-means (s)	Classical balanced k-means (s)	QUBO formulation (s)	Approximate embedding (s)
64	0.0218 ± 0.0017	0.0028 ± 0.0008	0.0008 ± 0.0003	0.1252
128	0.0256 ± 0.0022	0.0073 ± 0.0025	0.0070 ± 0.0008	0.4973
256	0.0334 ± 0.0035	0.0315 ± 0.0143	0.0192 ± 0.0018	1.9833
512	0.0414 ± 0.0060	0.1637 ± 0.0607	0.1154 ± 0.0024	7.9224
1024	0.0521 ± 0.0085	1.5577 ± 1.0501	0.4624 ± 0.0095	31.6696
2048	0.0684 ± 0.0134	10.8928 ± 5.5405	1.8409 ± 0.0201	126.6392
4096	0.1006 ± 0.0231	95.4876 ± 58.0103	7.6902 ± 0.0581	506.4798

We also report approximate embedding time for the quantum approach. All times are reported in seconds

Table 5 Time required to perform classical *k*-means, classical balanced *k*-means, and our QUBO formulation for data sets with an increasing number of clusters

Number of clusters	Classical *k*-means (s)	Classical balanced *k*-means (s)	QUBO formulation (s)	Approximate embedding (s)
2	0.02707 ± 0.0042	0.0276 ± 0.0101	0.0080 ± 0.0011	0.4973
4	0.0427 ± 0.0058	0.0417 ± 0.0164	0.0198 ± 0.0021	1.9833
8	0.0584 ± 0.0052	0.0399 ± 0.0113	0.1273 ± 0.0053	7.9224
16	0.0873 ± 0.0089	0.0390 ± 0.0094	0.5129 ± 0.0271	31.6696
32	0.1349 ± 0.0120	0.0271 ± 0.0052	1.9598 ± 0.0308	126.6392
64	0.2341 ± 0.0090	0.0201 ± 0.0023	7.6511 ± 0.0695	506.4798

We also report approximate embedding time for the quantum approach. All times are reported in seconds

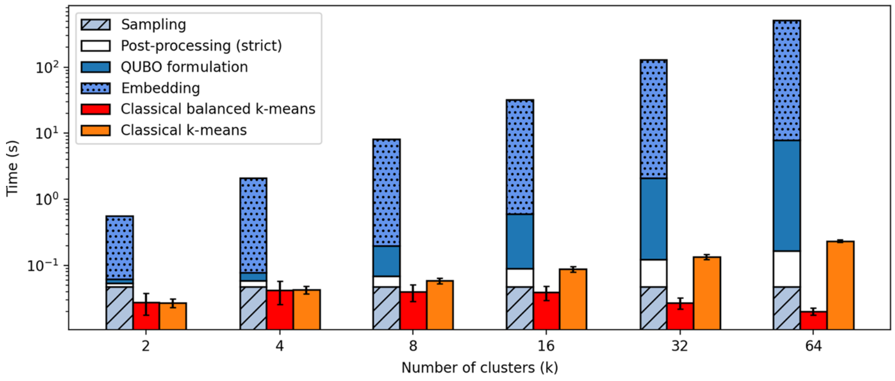

Fig. 9 Total computing time of quantum balanced k-means (blue bar), classical balanced k-means (red bar), and classical k-means (orange bar) as the number of clusters varies (k). Embedding and sampling times are approximate (Color figure online)

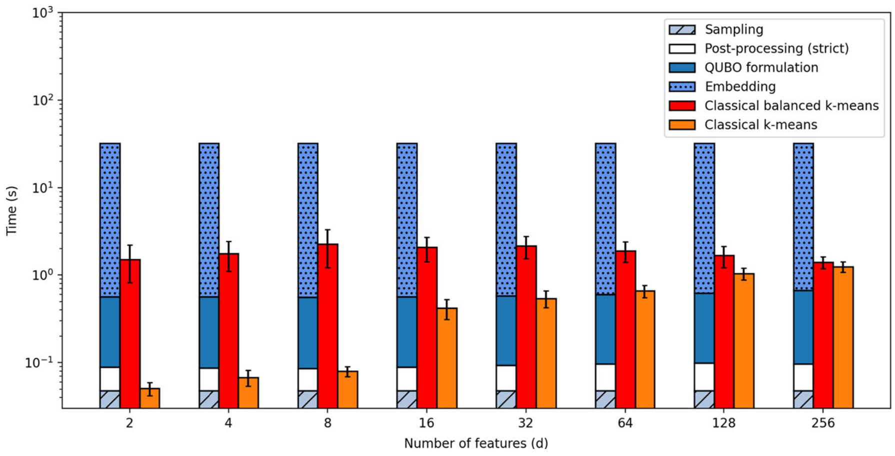

Fig. 10 Total computing time of quantum balanced k-means (blue bar), classical balanced k-means (red bar), and classical k-means (orange bar) as the number of features (d) varies. Embedding and sampling times are approximate (Color figure online)

4.2.9 Scalability with number of features (d)

Finally, we analyze the scalability of each algorithm with respect to the dimension of the training data set. For each problem type, all three clustering approaches were run on 50 synthetic classification data sets. Each data set consisted of $N = 1024$ points separated into $k = 4$ clusters. The average run time of each clustering approach is reported in Fig. 10 and Table 6.

As before, the quantum algorithm had the longest run time in all cases. However, on future hardware the quantum approach could perform better than classical k-means for $d \geq 128$ and better than classical balanced k-means for $d \leq 256$, depending on how well the embedding process is optimized. In Fig. 10, it appears that the quantum approach scales better than classical k-means as d increases. This is not surprising since

Table 6 Time required to perform classical *k*-means, classical balanced *k*-means, and our QUBO formulation on data sets with an increasing number of features

Number of features	Classical *k*-means (s)	Classical balanced *k*-means (s)	QUBO formulation (s)	Approximate embedding (s)
2	0.0508 ± 0.0089	1.5068 ± 0.6899	0.4742 ± 0.0185	31.6696
4	0.0681 ± 0.0137	1.7589 ± 0.6546	0.4771 ± 0.0189	31.6696
8	0.0803 ± 0.0105	2.2591 ± 1.0435	0.4737 ± 0.0101	31.6696
16	0.4190 ± 0.1065	2.0672 ± 0.6473	0.4760 ± 0.0102	31.6696
32	0.5411 ± 0.1171	2.1599 ± 0.6157	0.4895 ± 0.0179	31.6696
64	0.6598 ± 0.1048	1.8983 ± 0.4888	0.5033 ± 0.0178	31.6696
128	1.0369 ± 0.1577	1.6768 ± 0.4551	0.5283 ± 0.0221	31.6696
256	1.2474 ± 0.1726	1.4060 ± 0.2184	0.5759 ± 0.0207	31.6696

We also report approximate embedding time for the quantum approach. All times are reported in seconds

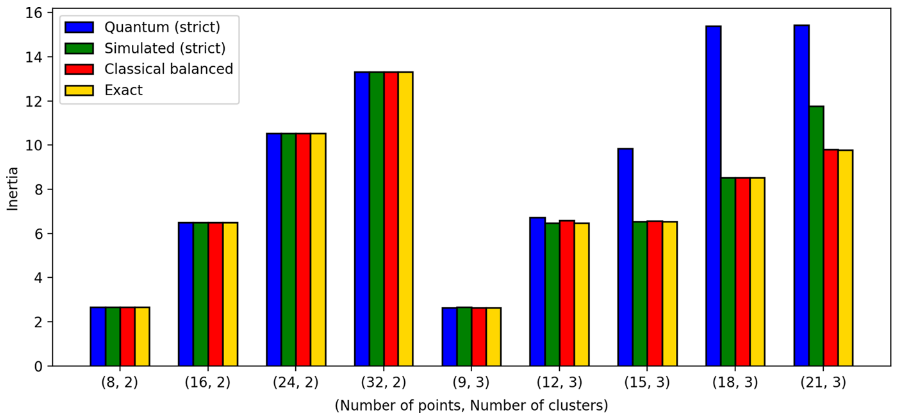

Fig. 11 Average inertia of all clustering approaches that guarantee equal size clusters. Exact data is reported in Table 7

the QUBO formulation only requires one computation related to the dimension of the data set (calculation of the distance matrix), while classical k-means requires distance calculations with each iteration. On the other hand, the quantum approach scales worse than classical balanced k-means. This is expected since the time complexity of classical balanced k-means is independent of d. It is somewhat surprising that the average run time of classical balanced k-means begins to decrease for $d > 32$, but we suspect this is due to cluster centers being farther apart on average.

4.3 Clustering a benchmark data set

As a final proof of concept, we clustered portions of the Iris benchmark data set using our quantum clustering approach. This data set contains 150 points (each with 4 features) divided into 3 equal-size classes. Unfortunately, due to qubit limitations on modern hardware, it is impossible to perform quantum balanced k-means clustering on the entire data set. Therefore, we generate smaller data sets by picking N/k points at random from $2 \leq k \leq 3$ of the data set's classes. When $k = 2$, all points were chosen from the first and second classes, which are linearly separable.

For a given problem type, all three clustering algorithms were run on 10 subsets of the Iris data set. For each trial, the quantum approach is performed once using quantum annealing on the D-Wave machine and once using D-Wave's simulated annealing sampler. Each returned QUBO solution is analyzed once using the “strict" post-processing algorithm and once using the “relaxed" post-processing algorithm. The α and β factors used for each problem type are reported in Table 1. Our results are reported in Table 7. Similar to Sect. 4.2.6, we also performed both post-processing algorithms on pseudo-solutions to the QUBO problem, consisting of Nk random bits. The inertia of the randomly generated solution was 2 to 3 times worse than the inertia of the worst clustering approach on average, so we have excluded random inertia values from our reported data.

Table 7 Average inertia values of the QUBO approach, classical balanced k-means, and classical k-means (Scikit-learn) when clustering subsets of the Iris data set

(N, k)	Quantum annealing (strict)	Simulated annealing (strict)	Classical balanced	Exact	Quantum annealing (relaxed)	Simulated annealing (relaxed)	Scikit-learn
(8, 2)	2.67	2.67	2.67	2.67	2.67	2.67	2.67
(16, 2)	6.50	6.50	6.50	6.50	6.50	6.50	6.50
(24, 2)	10.53	10.53	10.53	10.53	10.53	10.53	10.53
(32, 2)	13.32	13.32	13.32	13.32	13.32	13.32	13.32
(9, 3)	2.63	2.65	2.63	2.63	2.63	2.65	2.37
(12, 3)	6.73	6.48	6.59	6.48	6.73	6.48	5.21
(15, 3)	9.85	6.54	6.56	6.54	9.16	6.54	5.84
(18, 3)	15.38	8.52	8.53	8.52	13.73	8.52	7.55
(21, 3)	15.42	11.77	9.80	9.78	13.18	9.51	8.90

The QUBO approach is performed once with simulated annealing and once with quantum annealing. In each case, both post-processing algorithms are performed. The average inertia of the exact solution to the balanced clustering problem is included for reference

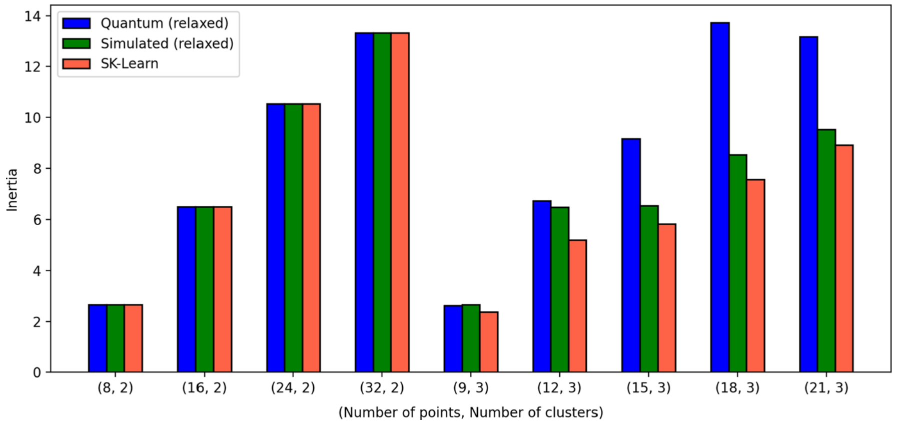

Fig. 12 Average inertia of all clustering approaches that do not guarantee equal size clusters. Exact data is reported in Table 7

First, consider the performance of the clustering approaches that guarantee equal size clusters (see Fig. 11). For all problems with $k = 2$, the QUBO approach (using simulated and quantum annealing) and the classical balanced algorithm correctly determined the exact solution. For problems with $k = 3$, the performance of the QUBO approach using quantum annealing begins to degrade as the number of points increases. However, the QUBO approach with simulated annealing continues to produce the exact solution for all problem types except (9, 3) and (21, 3). In fact, the QUBO approach with simulated annealing marginally outperforms the classical balanced approach for problems of type (12, 3), (15, 3), and (18, 3).

Next consider the performance of the clustering approaches that do not guarantee equal size clusters (see Fig. 12). Again, all clustering approaches correctly determined the exact solution to the balanced clustering problem for all problems where $k = 2$. Similar to Sect. 4.2.6, the Scikit-learn implementation better minimized inertia when $k = 3$, likely because generic k-means clustering avoids equal size clusters when advantageous. The most notable change when relaxing the constraint of equal size clusters is the difference in performance of simulated annealing for the problem type (21, 3). After relaxing the equal size clusters constraint, the simulated annealing approach achieves an inertia value less than that of the exact solution. The fact that the clustering assignments produced by simulated annealing were the same for all other problem types indicates that violations are still rare, even when relaxed post-processing is used.

We also determined the number of clusters assigned to each point and the number of points assigned to each cluster by the quantum computer before post-processing. We express this data in Table 8 following the same format used for the synthetic data sets in Sect. 4.2.6. The violations that occurred when clustering the Iris data set had many similarities with the violations that occurred when clustering synthetic data sets. Specifically, points were more commonly assigned to 0 clusters than 2 or more clusters, and clusters were more commonly assigned less than N/k points than more than N/k points. Again, as the number of points increased, violations became more prevalent.

Table 8 The frequency distribution of the number of clusters assigned to a point (expressed as a percentage of all points), and the frequency distribution of the number of points assigned to a cluster (expressed as a percentage of all clusters) when clustering subsets of the Iris data set

		Number of clusters assigned to each point				Number of points assigned to each cluster				
N	k	0	1	2	3	$n-3$	$n-2$	$n-1$	n	$n+1$
8	2	0.0	100.0	0.0	n/a	0.0	0.0	0.0	100.0	0.0
16	2	3.8	96.2	0.0	n/a	0.0	0.0	30.0	70.0	0.0
24	2	7.5	92.5	0.0	n/a	0.0	25.0	40.0	35.0	0.0
32	2	12.8	87.2	0.0	n/a	35.0	25.0	30.0	5.0	0.0
9	3	0.0	100.0	0.0	0.0	0.0	0.0	0.0	100.0	0.0
12	3	1.7	98.3	0.0	0.0	0.0	0.0	6.7	93.3	0.0
15	3	4.0	95.3	0.7	0.0	0.0	0.0	20.0	76.7	3.3
18	3	11.7	87.8	0.6	0.0	0.0	3.3	63.3	30.0	3.3
21	3	14.8	85.2	0.0	0.0	0.0	23.3	56.7	20.0	0.0

Situations that cannot occur, such as a point being assigned to more than k clusters or a cluster being assigned less than zero points, are denoted by “n/a." The expected number of points assigned to each cluster (N/k) is denoted by n

It is noteworthy that the QUBO approach performed better clustering subsets of the Iris data set in comparison to the synthetic data sets discussed in Sect. 4.2.6. The returned solutions better minimized inertia and violations were less common before post-processing. We believe this discrepancy occurs because the classes in the Iris data set are much more well defined that those in the synthetic data sets. With less well-defined classes, the quality of the quantum solution is more adversely impacted by the limited precision of the D-Wave machine.

5 Conclusion

As new applications of machine learning models continue to emerge, it is of great interest to improve upon existing training algorithms. Adiabatic quantum computers are a promising alternative platform for solving NP-hard or NP-complete training problems efficiently. In this paper, we propose a quantum approach to exactly solve the balanced k-means clustering training problem. We analyze our approach theoretically, and show that it scales favorably on large data sets when compared to current classical balanced k-means algorithms. We test our approach using the D-Wave 2000Q adiabatic quantum computer and compare it to the Scikit-learn implementation of classical k-means as well as our own implementation of the classical balanced k-means algorithm with the best time complexity. We demonstrated that on modern hardware, the clustering produced by our quantum approach is usually inferior to that produced by classical methods. However, we highlight that the solutions generated by our quantum approach exhibit some promising characteristics. As quantum hardware continues to improve in both fidelity and scale, we expect our approach to become a more viable alternative to existing classical balanced clustering algorithms.

In the future, we hope to test our approach on larger problems to gain a better understanding of how α and β scale with the number of points and number of clusters in the training data set. Additionally, we hope to generalize our QUBO formulation to satisfy the generic k-means clustering training problem (Problem 2). We also look to use elements of our approach to formulate quantum algorithms to similar clustering models, such as k-medoids clustering or fuzzy C-means clustering. Finally, we plan to investigate quantum approaches to clustering larger data sets within the qubit constraints of modern hardware.

Acknowledgements This manuscript has been authored by UT-Battelle, LLC under Contract No. DE-AC05-00OR22725 with the U.S. Department of Energy. The United States Government retains and the publisher, by accepting the article for publication, acknowledges that the United States Government retains a non-exclusive, paid-up, irrevocable, world-wide license to publish or reproduce the published form of this manuscript, or allow others to do so, for United States Government purposes. The Department of Energy will provide public access to these results of federally sponsored research in accordance with the DOE Public Access Plan (http://energy.gov/downloads/doe-public-access-plan).

Declarations

Conflict of interest The authors declare that they have no conflict of interest.

References

1. Aloise, D., Amit, D., Hansen, P., et al.: NP-hardness of Euclidean sum-of-squares clustering. Mach. Learn. (2009). https://doi.org/10.1007/s10994-009-5103-0
2. Blum, A., Rivest, R.L.: Training a 3-node neural network is NP-complete. In: Proceedings of the First Annual Workshop on Computational Learning Theory, pp. 9–18. Morgan Kaufmann Publishers Inc., Cambridge (1988). https://doi.org/10.1016/S0893-6080(05)80010-3
3. Hyafil, L., Rivest, R.: Constructing optimal binary decision trees is NP-complete. Inf. Process. Lett. (1976). https://doi.org/10.1016/0020-0190(76)90095-8
4. Willsch, D., Willsch, M., De Raedt, H., et al.: Support vector machines on the D-Wave quantum annealer. Comput. Phys. Commun. (2020). https://doi.org/10.1016/j.cpc.2019.107006
5. Dixit, V., Selvarajan, R., Alam, M.A., et al.: Training and classification using a restricted Boltzmann machine on the D-Wave 2000Q (2020). arXiv: 2005.03247
6. Date P., Schuman C., Patton R., et al.: A classical-quantum hybrid approach for unsupervised probabilistic machine learning. In: Advances in Information and Communication. FICC 2019. Lecture Notes in Networks and Systems, vol. 70. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-12385-7_9
7. Date, P., Potok, T.: Adiabatic quantum linear regression (2020). arXiv:2008.02355
8. O'Malley, D., Vesselinov, V., Alexandrov, B., et al.: Nonnegative/binary matrix factorization with a D-Wave quantum annealer. PLoS ONE (2018). https://doi.org/10.1371/journal.pone.0206653
9. Preskill, J.: Quantum computing in the NISQ era and beyond. Quantum **2**, 79 (2018)
10. Gupta, G., Younis, M.: Load-balanced clustering of wireless sensor networks. In: IEEE International Conference on Communications, pp. 1848-1852. IEEE (2003).https://doi.org/10.1109/ICC.2003.1203919
11. Ghosh, J., Strehl, A.: Clustering and visualization of retail market baskets. In: Pal, N.R., Jain, L. (eds.) Advanced Techniques in Knowledge Discovery and Data Mining. Advanced Information and Knowledge Processing. Springer, London (2005). https://doi.org/10.1007/1-84628-183-0_3
12. Banerjee, A., Ghosh, J.: Competitive learning mechanisms for scalable, incremental and balanced clustering of streaming texts. In: Proceedings of the International Joint Conference on Neural Networks, pp. 2697–2702. IEEE, Portland (2003). https://doi.org/10.1109/IJCNN.2003.1223993
13. Hartigan, J.A., Wong, M.A.: Algorithm AS 136: a k-means clustering algorithm. J. R. Stat. Soc.: Ser. C Appl. Stat. (1979). https://doi.org/10.2307/2346830
14. Arthur, D., Vassilvitskii, S.: How slow is the k-means method? In: Proceedings of the Twenty-Second Annual Symposium on Computational Geometry Association for Computing Machinery, pp. 144–153. ACM, New York (2006). https://doi.org/10.1145/1137856.1137880
15. Na, S., Xumin, L., Yong, G.: Research on k-means clustering algorithm: an improved k-means clustering algorithm. In: 2010 Third International Symposium on Intelligent Information Technology and Security Informatics, pp. 63–67 (2010). https://doi.org/10.1109/IITSI.2010.74
16. Celebi, M., Kingravi, H., Vela, P.: A comparative study of efficient initialization methods for the k-means clustering algorithm. Expert Syst. Appl. (2013). https://doi.org/10.1016/j.eswa.2012.07.021
17. Kapoor, A., Singhal, A.: A comparative study of k-means, k-means++ and fuzzy c-means clustering algorithms. In: 2017 3rd International Conference on Computational Intelligence Communication Technology (CICT), pp. 1–6 (2017). https://doi.org/10.1109/CIACT.2017.7977272
18. Pedregosa, F., Varoquaux, G., Gramfort, A., et al.: Scikit-learn: machine learning in Python. J. Mach. Learn. Res. (2011). https://doi.org/10.5555/1953048.2078195
19. Bennett, K.P., Bradley, P.S., Demiriz, A.: Constrained k-means clustering (2000)
20. Ganganath, N., Cheng, C., Tse, C.K.: Data clustering with cluster size constraints using a modified k-Means algorithm. In: 2014 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery, pp. 158–161. IEEE, China (2014). https://doi.org/10.1109/CyberC.2014.36
21. Malinen, M.I., Fränti, P.: Balanced k-means for clustering. In: Fränti, P., Brown, G., Loog, M., Escolano, F., Pelillo, M. (eds.) Structural, Syntactic, and Statistical Pattern Recognition, pp. 32–41. Springer, Berlin (2014). https://doi.org/10.1007/978-3-662-44415-3_4
22. Khan, S.U., Awan, A.J., Vall-Llosera, G.: K-Means clustering on noisy intermediate scale quantum computers (2019). arXiv:1909.12183
23. Ushijima-Mwesigwa, H., Negre, C., Mniszewski, S.: Graph partitioning using quantum annealing on the D-Wave system. In: Proceedings of the Second International Workshop on Post Moores Era Supercomputing, pp. 22–29. ACM, New York (2017)

24. Neukart, F., Dollen, D., Seidel, C.: Quantum-assisted cluster analysis. Front. Phys. (2018). https://doi.org/10.3389/fphy.2018.00055
25. Wereszczyński, K., Michalczuk, A., Josiński, H., et al.: Quantum computing for clustering big datasets. In: 2018 Applications of Electromagnetics in Modern Techniques and Medicine (PTZE), pp. 276–280 (2018). https://doi.org/10.1109/PTZE.2018.8503109
26. Bauckhage, C., Ojeda, C., Sifa, R., Wrobel, S.: Adiabatic quantum computing for kernel k = 2 means clustering. In: LWDA (2018)
27. Bauckhage, C., Piatkowski, N, Sifa, R., et al.: A QUBO formulation of the k-Medoids Problem. In: LWDA (2019)
28. Kumar, V., Bass, G., Tomlin, C., et al.: Quantum annealing for combinatorial clustering. Quantum Inf. Process. (2018). https://doi.org/10.1007/s11128-017-1809-2
29. Date, P., Arthur, D., Lauren, P.: QUBO formulations for training machine learning models. Sci. Rep. (2021). https://doi.org/10.1038/s41598-021-89461-4
30. Date, P., Patton, R., Schuman, C., et al.: Efficiently embedding QUBO problems on adiabatic quantum computers. Quantum Inf. Process. **18**, 117 (2019). https://doi.org/10.1007/s11128-019-2236-3
31. Inaba, M., Katoh, N., Imai, H.: Applications of weighted Voronoi diagrams and randomization to variance-based k-clustering. In: Proceedings of the Tenth Annual Symposium on Computational Geometry, pp. 332–339. ACM, New York (1994). https://doi.org/10.1145/177424.178042