A Quantum Annealing-Based Approach to Extreme Clustering

Tim Jaschek\textsuperscript{1,2,*} and Jaspreet S. Oberoi\textsuperscript{1,3,*}

\textsuperscript{1}1QB Information Technologies (1QBit), Vancouver, BC, Canada
\textsuperscript{2}Department of Mathematics, University of British Columbia, Vancouver, BC, Canada
\textsuperscript{3}School of Engineering Science, Simon Fraser University, Burnaby, BC, Canada

\textsuperscript{*}Both authors have contributed equally to the manuscript.

\textbf{Abstract}—In this age of data abundance, there is a growing need for algorithms and techniques for clustering big data in an accurate and efficient manner. Well-known clustering methods of the past are computationally expensive, especially when employed to cluster massive datasets into a relatively large number of groups. The particular task of clustering millions (billions) of data points into thousands (millions) of clusters is referred to as extreme clustering. We have devised a distributed method, capable of being powered by a quantum processor, to tackle this clustering problem.

\textbf{Index Terms}—extreme clustering, distributed computing, quantum computing, maximum weighted independent set, unsupervised learning, data mining

I. INTRODUCTION

The worldwide explosion in the availability of data has raised opportunities and challenges alike on how best to make use of it. Those who are able to pose the right questions in exploring datasets and to understand the high-value answers that may be found therein have experienced a significant competitive advantage. Clustering is an established technique for grouping dataset elements based on similarity, and helps data practitioners execute required analyses, sometimes independently and at other times by acting as a precursor to various data mining and machine learning algorithms.

Traditionally, clustering approaches have been built, customized, and used for tasks where the resultant number of clusters is not significantly high. Clustering the customers of a supermarket chain into a small number of groups, curating a news feed by clustering articles into five to ten categories, or conducting inventory management by identifying ten to twenty product groupings are just a few such examples. Algorithms such as \textit{k}-means, BIRCH [37], and spectral clustering perform very well on such tasks and give high-quality solutions in a reasonable runtime. This is because these traditional algorithms scale reasonably well with respect to the number of data points \( n \). However, in most cases, the dependence of the computational complexity of these algorithms on the number of clusters \( k \) is either exponential or higher-order polynomial, and thus they fail to keep the computation time under control for large values of the parameter \( k \). Another common problem is that some of these algorithms require vast amounts of memory.

The demand for clustering algorithms has evolved beyond a small value of \( k \). Present-day clustering examples involve deciphering the content of billions of web pages by grouping them into millions of topics (or “labels”) \([8, 30]\), identifying duplicates among billions of images using nearest-neighbour detection \([23, 34]\), and clustering hundreds of billions of satellite images into a few billion categories \([36]\). This domain of clustering, where \( n \) and \( k \) are both substantially large numbers, has been referred to as extreme clustering in a previous work \([18]\). Although there is substantial value in perfecting this type of clustering, very little effort has been made by the machine learning community \([18, 36]\).

Our algorithm is, in fact, such an effort in the extreme clustering space. The output of our algorithm is a clustering tree, which can be used to generated multiple clustering assignments (or “levels”) with varying degrees of accuracy. Generating such a tree is not uncommon for clustering algorithms. Consider, for example, hierarchical clustering algorithms which generate binary clustering trees. While higher levels in the tree correspond to clustering assignments consisting of multiple small clusters, lower levels correspond to fewer (yet larger) clusters. As such, a clustering tree is a useful tool for dealing with real-world data visualization problems. Our algorithm BiDViT, short for “Big Data Visualization Tool”, provides this functionality.

A remarkable feature of the proposed approach is its ability to harness the potential of quantum annealing. In the past years, quantum annealing and simulated annealing have been applied to clustering problems. Consider, for example, \([20]\) and \([29]\), where quantum annealing is used to solve combinatorial clustering problems similar to the \( k \)-means clustering problem, and \([21]\), which studies the application of quantum annealing for probabilistic clustering models such as Gaussian mixture models and latent Dirichlet allocation. However, the mentioned approaches are not capable of addressing problems in the extreme clustering domain.

This paper is organized as follows. We present the high-level features of our algorithm in Section II. In Section III, we describe the core problem that our algorithm solves, and prove that, under a separability assumption, our coarsening method yields an optimal clustering assignment. Section IV contains details on the mathematical reasoning behind our approach and insights to its implementation. After emphasizing its relationship to the maximum weighted independent set problem (Section V) and carrying out a complexity analysis (Section VI), we provide results on the runtime and quality of the solutions obtained using our algorithm in comparison to state-of-the-art clustering algorithms.
II. AN APPROACH TO EXTREME CLUSTERING

Our algorithm is based on a distance-based clustering method, which we call coarsening and describe in Section III. Our method does not require the number of clusters \( k \) as an input parameter, but instead is driven by a parameter which we call the \textit{radius of interest} and denote by \( \varepsilon \). The computational complexity of our method does not depend on this parameter.

In general, the computational complexity of distance-based clustering methods is proportional to the square of the dataset cardinality, because all pairwise distances need to be computed. We overcome this bottleneck by employing a \textit{divide-and-conquer} approach. The idea behind this approach is to intelligently divide the original dataset into smaller data \textit{chunks}, and then process these chunks independently. This yields a result different from that which we would obtain when applying our coarsening scheme on the entire dataset; however, its slight imprecision results in a significant computational speed-up.

The number of data points a user desires to have in a chunk is referred to as \textit{chunk cardinality} and denoted by \( \kappa \). A user should make the determination of \( \kappa \) by taking into account the number of available parallel processors, the data handling capacity of each processor, or, in the case of using a quantum annealer, the number of available fully connected qubits. For example, if a user plans to use the D-Wave 2000Q quantum annealer \cite{6} to process a data chunk, the user might wish to fix the chunk cardinality to be around 2000 data points. In such a case, a dataset with 512,000 points will need to be divided into 256 chunks. Fujitsu’s second-generation “Digital Annealer Unit” \cite{10} provides 8192 fully connected qubits and thus would require about 64 chunks in the same scenario. Since the coarsening procedure for each chunk can be executed independently, our algorithm can make use of multiprocessing and distributed computing.

III. THE COARSENING METHOD

The key idea of our coarsening method is to approximate a finite set \( X \subset \mathbb{R}^d \) by a subset \( S \subset X \) such that, for any point \( x \in X \), there exists \( y \in S \) such that \( d(x,y) < \varepsilon \), for some parameter \( \varepsilon > 0 \). Here, \( d \) denotes a metric on \( \mathbb{R}^d \), for example, the Euclidean distance given by \( d(x,y) = \| x - y \|_2 \). In this case, we say that \( S \) is \( \varepsilon \)-dense in \( X \) and call \( \varepsilon \) the \textit{radius of interest}. Note that this concept is not restricted to subsets of Euclidean spaces and can be generalized to an arbitrary metric space \((M,d)\). In what follows, we will assume that \( X = \{x^{(1)}, \ldots, x^{(n)}\} \) is a dataset consisting of \( n \) \( d \)-dimensional data points. Finding an arbitrary \( \varepsilon \)-dense subset of \( X \) does not necessarily provide us with a helpful approximation. For example, \( X \) itself is always \( \varepsilon \)-dense in \( X \). However, enforcing the additional constraint that any two points in the subset \( S \) have to be separated by a distance \( \geq \varepsilon \) yields more-interesting approximations, which often lead to a reduction in the number of data points (one of our primary objectives). We will call such a set \( \varepsilon \)-separated.

Figure 1 shows an example of a point cloud and a \( \varepsilon \)-dense, \( \varepsilon \)-separated subset.

The following theorem shows that a maximal \( \varepsilon \)-separated set \( S \) of \( X \) is necessarily \( \varepsilon \)-dense in \( X \).

\textbf{Theorem 1.} Let \( S \) be a maximal \( \varepsilon \)-separated subset of \( X \) in the sense of set inclusion, that is, there exists no set \( S' \) such that \( S \subsetneq S' \) and \( S' \) is \( \varepsilon \)-separated. Then \( S \) is \( \varepsilon \)-dense in \( X \). Furthermore, \( S \) is a minimal \( \varepsilon \)-dense set, that is, there exists no set \( S' \) such that \( S \subsetneq S' \) and \( S' \) is \( \varepsilon \)-dense in \( X \).

\textit{Proof.} Let \( S \) be as above and assume in contradiction that \( S \) is not \( \varepsilon \)-dense in \( X \). Then we could find \( x \in X \) such that \( d(x,y) \geq \varepsilon \), for every \( y \in S \). Hence, \( S \cup \{x\} \) is \( \varepsilon \)-separated, which is in contradiction to the maximality of \( S \).

To prove the second statement, we fix a point \( x \in S \). Since \( S \) is \( \varepsilon \)-separated, \( d(x,y) \geq \varepsilon \) for any \( y \in S \) and, thus, \( S \setminus \{x\} \) is not \( \varepsilon \)-dense in \( X \).

Note that a maximal \( \varepsilon \)-separated subset does not refer to a \( \varepsilon \)-separated subset with fewer or equally many elements than all other \( \varepsilon \)-separated subsets but, rather, to an \( \varepsilon \)-separated subset that is no longer \( \varepsilon \)-separated when adding a single data point.

Contrary to the statement of \textbf{Theorem 1}, a minimal \( \varepsilon \)-dense subset does not need to be \( \varepsilon \)-separated. Consider the set \( X = \{1,2,3,4\} \subset \mathbb{R} \), where \( X \subset \mathbb{R} \) indicates that we are referring to the Euclidean distance. Then, \( S = \{2,3\} \) is \( 3/2 \)-dense in \( X \) but not \( 3/2 \)-separated.

In the following, we assume that \( X \) is equipped with a \textit{weight function} \( w: X \rightarrow \mathbb{R}_+ \). We call \( w_i = w(x^{(i)}) \) the weight of \( x^{(i)} \) and summarize all weights in a \textit{weight vector} \( w \in \mathbb{R}^d_+ \). It will be clear from the context whether we refer to a weight function or a weight vector. The weight of a set \( S \subset X \) is given by \( \omega(S) = \sum_{x \in S} w(x) \). We are interested in solving the following optimization problem:

\[
\begin{align*}
\text{maximize} & \quad \omega(S) \\
\text{subject to} & \quad S \subset X \\
& \quad \text{is } \varepsilon \text{-separated.}
\end{align*}
\] (1)

If we impose unit weights, the solution set to this optimization problem will consist of the maximal \( \varepsilon \)-separated subsets.
of $X$ with a maximum number of elements among all such subsets. The term “maximal” refers to set inclusion and the “maximum” refers to set cardinality.

Since $w(x) > 0$ for all $x \in X$, a solution $S^*$ to Problem (1) will always be a maximal $\epsilon$-separated subset and, therefore, by Theorem 1, $\epsilon$-dense. In Section V, we show that this problem is equivalent to solving a maximum weighted independent set (MWIS) problem for a graph depending on $X$ and $\epsilon$. Thus, the computational task of finding a maximal $\epsilon$-separated subset of maximum weight is NP-hard [17, 26].

Note that an $\epsilon$-separated subset $S$ gives rise to a clustering assignment $C = \{C_1, \ldots, C|S|\}$. This assignment is given by

$$C_i = \{y \in X : d(x, y) < d(x', y) \text{ for all } x' \in S\}. \quad (2)$$

More details on the clustering assignment are provided in Section IV. A common assumption in clustering literature is separability—not to be mistaken with $\epsilon$-separability—of the dataset with respect to a clustering $C$. The dataset $X$ is called separable with respect to a clustering $C = \{C_1, \ldots, C_k\}$ if

$$\max_{x, y \in C_i, 1 \leq i \leq k} d(x, y) \leq \min_{x, y \in C_i \cup C_j, 1 \leq i \neq j \leq k} d(x, y),$$

that is, if the maximum intra-cluster distances are smaller than the minimum inter-cluster distances. The following theorem shows that, if $\epsilon$ is chosen correctly, our coarsening method yields the true clustering assignment.

**Theorem 2.** Let $X$ be separable with respect to a clustering $C = \{C_1, \ldots, C_k\}$. Then, for any $\epsilon \in \left(\max_{x, y \in C_i, 1 \leq i \leq k} d(x, y), \min_{x, y \in C_i \cup C_j, 1 \leq i \neq j \leq k} d(x, y)\right)$, the coarsening yields the correct clustering assignment.

**Proof.** First note that, by the separability assumption, the above interval is non-empty. Next, one can see that, for any admissible choice of $\epsilon$, any two points from different clusters are $\epsilon$-separated. Indeed, for $x \in C$ and $y \in C'$, it holds that

$$d(x, y) \geq \min_{x', y' \in C_i \cup C_j, 1 \leq i \neq j \leq k} d(x', y') \geq \epsilon.$$

Furthermore, if a point $x$ in a cluster $C$ is selected, then no other point $y$ in the same cluster can be selected because

$$d(x, y) \leq \max_{x', y' \in C_i, 1 \leq i \leq k} d(x', y') < \epsilon.$$

Therefore, every solution $S \subset X$ to Problem (1) is a union of exactly one point from each cluster. Using the separability of $X$ with respect to $C$, we can see that the clustering assignment induced by (2) is coincident with $C$. \hfill $\Box$

In practice, the separability assumption is rarely satisfied, and it is challenging to select $\epsilon$ as above (as this assumes some knowledge about the clustering assignment). However, Theorem 2 shows that our coarsening method is of interest, and can potentially yield optimal clustering assignments.

We have developed two separate methods to address the NP-hard task of solving problem Problem (1). One method, which we will refer to as the heuristic method, loosens the condition of having a maximum weight. The heuristic method constructs a single maximal $\epsilon$-separated subset by picking a root $x_0 \in X$ and successively adding elements $x_{k+1}$ that are at least a distance of $\epsilon$ away from all previously added elements $x_0, \ldots, x_k$ until no such element $x_{k+1}$ exists. All of the elements are selected from among all candidates of maximum weight, following a greedy strategy. The second method, which we will refer to as the quantum method, explores all different maximal $\epsilon$-separated subsets simultaneously, yielding one that has maximum weight. The quantum method is based on the formulation of a quadratic unconstrained binary optimization (QUBO) problem, which can be solved efficiently using a quantum annealer such as the D-Wave 2000Q or a digital annealer such as the one developed by Fujitsu.

Both our methods can be used to construct $\epsilon$-nets of metric measure spaces $(M, d, \mu)$. Such spaces are of interest in diverse areas of mathematics, such as differential geometry and probability theory, as they provide roughly isometric approximations of the original space [5, 16].

**IV. Algorithm**

In this section, we explain the BiDVIT algorithm in detail. The coarsening method explained earlier is a part of this algorithm. Let $X = \{x^{(1)}, \ldots, x^{(n)}\} \subset \mathbb{R}^d$ denote a dataset of a $d$-dimensional data points. Note that, mathematically speaking, a dataset is not a set but rather a multiset, that is, we allow repetitions. Our algorithm consists of two parts: data partitioning and data coarsening. The data coarsening can be further subdivided into chunk coarsening and chunk collapsing.

**A. Data partitioning**

Dividing the original dataset into smaller datasets is called partitioning. A partition $P$ of $X$ is a collection of non-empty disjoint subsets $P_1, \ldots, P_k \subset X$ such that $X = \bigcup_{P \in P} P$. Elements of partitions are typically referred to as blocks, parts, or cells; however, we refer to them as chunks throughout this paper. Our partitioning procedure can be interpreted as extracting hyper-cuboids from a $d$-dimensional space, where $d$ is the dimension of the dataset. This extraction is intended to be homogeneous, meaning that every extracted chunk has an equal number of data points, that is, a constant chunk cardinality (there might be minor differences when the cardinality of the chunk to be divided is odd). The homogeneity of the chunks is important for two reasons. First, it makes it possible for a fixed-sized processor to process the entire set of chunks, one after another, without having to worry about any of the chunks being over-sized and thus not able to be processed. Second, it helps the BiDVIT algorithm to better capture the data distribution. We use a heuristic technique for data partitioning, which is inspired by the binary space partitioning method used for recursively subdividing a space into convex sets by hyperplanes.
To divide the data, we pick an axis of maximum variance (such an axis might not be unique) uniformly at random among the elements of the set
\[
\arg \max_{1 \leq j \leq d} \{ \sigma_j^2 \},
\]
where \( \sigma_j^2 = \frac{\sum (x_j - \mu_j)^2}{n} \) and \( \mu_j = \frac{\sum x_j}{n} \) are defined as usual. We then split the data along the axis that we selected, say \( \ell \), at the median \( m \) of \( \{x^{(1)}_\ell, ..., x^{(n)}_\ell\} \) in a way such that we obtain two data chunks \( P_1 \) and \( P_2 \), whose cardinalities differ by at most one (for the case where \( n \) is odd) and which satisfy
\[
P_1 \subseteq \{x \in X : x_\ell \leq m\} \quad \text{and} \quad P_2 \subseteq \{x \in X : x_\ell \geq m\}.
\]
Note that we cannot simply assign \( P_1 = \{x \in X : x_\ell \leq m\} \) and \( P_2 = X \setminus P_1 \) because these sets might differ drastically in their cardinality. For example, when \( x^{(1)}_\ell = ... = x^{(n)}_\ell \), this assignment would imply \( P_1 = X \) and \( P_2 = \emptyset \).

By using \( P_1 \) and \( P_2 \) in the role of \( X \), this process can be repeated iteratively and would yield a binary tree of data chunks. After \( s \) iterations, this would leave us with 2\( s \) chunks \( P^{(s)}_k \) such that
\[
X = \bigcup_{1 \leq k \leq 2^s} P^{(s)}_k,
\]
where the union is disjoint. In our specific application, we perform partitioning steps until the number of data points in the chunk to be divided is less than or equal to the chunk cardinality parameter \( \kappa \). Figure 2 provides a visualization.

**B. Chunk Coarsening**

After dividing the high-volume data into chunks, each of which now holds a manageable amount of data, the algorithm progresses to execute data abstraction, chunk by chunk. The data coarsening technique that we propose is independent for each chunk and the coarsening can thus be performed in parallel. The goal of a data coarsening step is, for each chunk, to find a set of representative data points such that the resultant representative point cloud, having fewer data points, can replace the original point cloud, while still representing the original data distribution in a faithful manner. Rather than depicting a single data point, each representative point is intended to depict multiple data points (in close proximity) belonging to the original point cloud.

Let \( x^{(1)}, ..., x^{(n)} \) denote the data points within a chunk \( P \) and \( \varepsilon > 0 \) be the radius of interest, which depicts the extent of similarity. In what follows, we assume that all the data points are pairwise different. Practically, this can be achieved by removing duplicates and cumulatively incrementing the weight of the representative point we wish to keep by the weight of the discarded duplicates. The radius of interest \( \varepsilon \) induces a weighted graph \( G^\varepsilon = (P, E^\varepsilon, w_P) \), where the vertices correspond to the data points \( P \), the edge set \( E^\varepsilon \) is given by the relation \( \sim_\varepsilon \) defined by
\[
x^{(i)} \sim_\varepsilon x^{(j)} \iff d(x^{(i)}, x^{(j)}) < \varepsilon,
\]
and the weight function \( w_P : P \rightarrow \mathbb{R}_+ \) is the restriction of \( w \) to \( P \). Due to the absence of duplicates and the triangle inequality, \( \sim_\varepsilon \) is an equivalence relation on chunk \( P \). For each data point \( x^{(i)} \) we denote its weight \( w_P(x^{(i)}) \) by \( w_i \). If \( x^{(i)} \sim_\varepsilon x^{(j)} \), we say that \( x^{(i)} \) represents \( x^{(j)} \) or that \( x^{(i)} \) and \( x^{(j)} \) are neighbours. Clearly, this implies that \( x^{(i)} \) is an element of the open metric circle (in the 2D case) or the open metric ball (in the case of arbitrary dimensions) of radius \( \varepsilon \) centred at \( x^{(i)} \), which we denote by \( B(x^{(i)}, \varepsilon) \). We say that a set \( S \subseteq P \) represents the chunk \( P \), if for every \( x \in P \) there exists a vertex \( y \in S \) such that \( y \) represents \( x \), that is, if \( S \) is \( \varepsilon \)-dense in \( P \).

As described in Section III, our goal is to find an \( \varepsilon \)-separated subset \( S \) of chunk \( P \) with maximum weight. For each data point \( x^{(i)} \), we introduce a binary decision variable \( s_i \) that encodes whether \( x^{(i)} \) is used in a possible set \( S \).

Furthermore, we define the neighbourhood matrix \( N(x) \) of the graph \( G^\varepsilon = (P, E^\varepsilon, w_P) \) by
\[
N_{ij} = \begin{cases} 1, & x^{(i)} \sim_\varepsilon x^{(j)}; \\ 0, & \text{otherwise}. \end{cases}
\]

The problem can then be posed as a quadratically constrained quadratic program (QCQP)

\[
\begin{align*}
\text{maximize} & \quad \sum_{i=1}^n s_i w_i \\
\text{subject to} & \quad \sum_{i=1}^n \sum_{j>i} s_i N_{ij} s_j = 0.
\end{align*}
\]

(3)

Here, the inner summation of the constraint does not need to run over all indices due to the symmetry of \( N(x) \). The matrix form of Problem (3) is given by

\[
\begin{align*}
\text{maximize} & \quad s^T w \\
\text{subject to} & \quad s^T N(x) s = 0,
\end{align*}
\]

where \( N(x) \) is the upper diagonal matrix of \( N(x) \). There exists a wide range of software solvers for QCQPs but, as we will see in Section V, Problem (3) is equivalent to the NP-hard MWIS problem for \( G^\varepsilon = (P, E^\varepsilon, w_P) \), and thus is computationally intractable for large problem sizes. We now explain two methods we have devised for addressing Problem (3).
1) The Heuristic Method: We wish to emphasize that the heuristic method does not provide us with a solution to Problem (3). Rather, the aim of this method is to obtain higher weights in subsequent steps. Figure 3 provides an example.

Choose \( x^{(i)} \) has the consequence that the data points \( x^{(i)} \) with \( x^{(i)} \sim_{\epsilon} x^{(i)} \) can no longer be elements of the set \( S^* \).

Hence, the next index \( \ell_2 \) must be selected from the set \( \{ 1 \leq i \leq n \} \). In general, having chosen \( k \) elements \( \ell_1, \ldots, \ell_k \), we select the \((k+1)\)-th element uniformly at random such that

\[
\ell_{k+1} = \arg \max_{i \in I_k} \{ w_i \},
\]

where

\[
I_k = \{ 1 \leq i \leq n : x^{(i)} \sim_{\epsilon} x^{(i)}, \ell = \ell_1, \ldots, \ell_k \}.
\]

This procedure can be repeated iteratively until \( I_k = \emptyset \). Algorithm 1 represents our heuristic method in pseudocode form.

The heuristic method is a greedy approach to Problem (3). In each step, we add the data point which is the local best choice. The method does not take into account that it might be beneficial to start with a data point with a lower weight, which might lead to situations where we could possibly select higher weights in subsequent steps. Figure 3 provides an example.

2) The Quantum Method: In contrast to the heuristic method, the QUBO approach provides a feasible (i.e., non-approximate) solution Problem (3). We reformulate the problem by transforming the QCQP into a QUBO problem.

Using the Lagrangian penalty method, we incorporate the constraint into the objective function by adding a penalty term. Let \( \lambda > 0 \) be a penalty multiplier, the value of which we will clarify momentarily. For large enough \( \lambda \), the solution set of Problem (3) is equivalent to that of the QUBO problem

\[
\max_{s \in \{0,1\}^n} \sum_{i=1}^n s_i w_i - \lambda \sum_{i=1,j=i+1}^n s_i N_{ij}^{(s)} s_j.
\]

Writing this as a minimization problem instead of a maximization problem and using matrix notation yields

\[
\min_{s \in \{0,1\}^n} \ s^T Q s,
\]

where

\[
Q_{ij} = \begin{cases} 
-w_i, & i = j, \\
\lambda, & N_{ij}^{(s)} = 1 \text{ and } i < j, \\
0, & \text{otherwise}.
\end{cases}
\]

It is worth noting that the number of variables in this QUBO formulation is equal to the number of data points to be coarsened. Another advantage of this problem formulation is the existence of tight bounds on the penalty parameter \( \lambda \). The following theorems show that Problem (3) is equivalent to Problem (4) for every penalty parameter \( \lambda > \max_{1 \leq i \leq n} w_i \). When all the weights are assumed to be positive, the term on the right-hand side coincides with \( \|w\|_\infty \).

**Theorem 3.** Let the penalty multiplier \( \lambda \) satisfy \( \lambda > \|w\|_\infty \). Then, for any solution \( s \in \{0,1\}^n \) to the QUBO Problem (4), the corresponding set \( S = \{ x^{(i)} : s_i = 1, i = 1, \ldots, n \} \) is \( \varepsilon \)-separated.

**Proof.** We follow [1] and show that every solution \( s \) to Problem (4) satisfies the separation constraint

\[
\sum_{i=1}^n \sum_{j=i+1}^n s_i N_{ij}^{(s)} s_j = 0.
\]

Assuming, in contradiction, that the opposite were to be the case, that is, we could find a solution \( s \) and indices \( k \) and \( \ell \) such that \( 1 \leq k < \ell \leq n \) and \( s_k = s_\ell = N_{k\ell}^{(s)} = 1 \). Let \( e_k \) denote the \( k \)-th standard unit vector, and define \( v = s - e_k \). Then matrix algebra shows

\[
v^T Q v = s^T Q s - e_k^T Q s - e_k^T Q e_k
\]

\[
= s^T Q s - \sum_{j=1}^n s_j Q_{jk} - \sum_{i=k}^n s_i Q_{ik} - Q_{kk}
\]

\[
= s^T Q s - \lambda \sum_{i \neq k} s_i N_{ik}^{(s)} + w_k.
\]

Since \( N_{kk}^{(s)} = s_\ell = 1 \), we have \( \sum_{i \neq k} s_i N_{ik}^{(s)} \geq 1 \) and thus

\[
v^T Q v < s^T Q s - \lambda + w_k.
\]

Therefore, as \( \lambda > \|w\|_\infty \), it holds that \( v^T Q v < s^T Q s \), which is absurd since, by assumption, \( s \) solves Problem (4). □

---

**Algorithm 1:** Heuristic Method

```
input : \( P = \{ x^{(1)}, \ldots, x^{(n)} \} \) is a data chunk; \( w \) is a weight vector; \( \varepsilon \) is a radius of interest
output: \( S^* \) is a maximal \( \varepsilon \)-separated subset with high (but not necessarily highest) weight
1 \( I \leftarrow \{ 1, \ldots, n \} \)
2 \( S^* \leftarrow \emptyset \)
3 compute the \( \varepsilon \)-neighbourhood matrix \( N^{(\varepsilon)} \)
4 while \( I \neq \emptyset \) do
5 \( \ell \in \arg \max_{j \in I} \{ w_j \} \) uniformly at random
6 \( I \leftarrow \{ j \in I : N_{\ell,j}^{(\varepsilon)} = 0 \} \)
7 \( S^* \leftarrow S^* \cup \{ x^{(\ell)} \} \)
8 end
9 return \( S^* \)
```
Theorem 1 and Theorem 3 imply that a solution to Problem (4) is $\varepsilon$-dense. Furthermore, one can show that, under the assumption of Theorem 3, the solution sets of Problem (3) and Problem (4) are equivalent.

**Theorem 4.** Let the penalty multiplier $\lambda$ satisfy $\lambda > \|w\|_\infty$. Then, the solution sets of Problem (3) and Problem (4) are equivalent.

**Proof.** Note that Problem (3) is equivalent to

$$\begin{align*}
\text{minimize } & -s^T w \\
\text{subject to } & s^T N(\varepsilon) s = 0.
\end{align*} \tag{5}$$

Let $s_1$ and $s_2$ be solutions to Problem (5) and Problem (4), respectively. Denote the objective functions by $p_1(s) = -s^T w$ and $p_2(s) = -s^T w + \lambda (s^T N(\varepsilon) s)$. Then, since $\lambda > \|w\|_\infty$, by Theorem 3, both $s_1$ and $s_2$ satisfy the separation constraint. Using the fact that $s_1$ and $s_2$ minimize $p_1$ and $p_2$, respectively, we can see that

$$p_1(s_1) \leq p_1(s_2) = p_2(s_2) \leq p_2(s_1) = p_1(s_1).$$

Hence, the inequalities must actually be equalities; thus, the solution sets of the optimization problems coincide.

Closely following the proof of Theorem 3, one can see that, for the selected indices $1 \leq k < \ell \leq n$, all we use is the inequality $\lambda > \max\{w_k, w_\ell\}$. Therefore, by using individual penalty multipliers instead of a single multiplier, we can minimize the QUBO polynomial described by the matrix

$$Q_{ij} = \begin{cases} -w_i, & i = j, \\
\lambda_{ij}, & N_{ij}(\varepsilon) = 1 \text{ and } i < j, \\
0, & \text{otherwise},
\end{cases}$$

where $\lambda_{ij} > \max\{w_i, w_j\}$. This slight modification in the QUBO formulation has the advantage that most of the coefficients are smaller, which leads to improved performance when solving the problem using a quantum annealer.

In the absence of a quantum annealer, solutions to QUBO problems can be approximated using heuristics such as path relinking [25], tabu search [25], and parallel tempering [38].

Figure 3 shows a graph where the quality of the result of the heuristic method is worse than that of quantum method.

**C. Chunk Collapsing**

Having identified a maximal $\varepsilon$-separated subset $S \subseteq P$, we collapse the vertices $P \setminus S$ into $S$. By “collapsing” we mean updating the weight of each $x \in S$ according to the weights of all $y \in P \setminus S$ that satisfy $x \sim_\varepsilon y$. We aim to assign each $y \in P \setminus S$ to a unique $x \in S$. To do so, we generate a so-called Voronoi decomposition of each chunk $P$, which is a partition, where each point $x \in P$ is assigned to the closest point within a subset $S$. In our case, “close” refers to the Euclidean distance, but one could use different metrics, for example, the ones induced by the $\ell_1$- or $\ell_\infty$-norm. A Voronoi partition is depicted in Figure 4.

To obtain such a partition, we define the sets

$$A_x = \{y \in X : d(x, y) \leq d(x', y) \text{ for all } x' \in S\},$$

for each $x \in S$. By construction, $A_x$ contains all vertices that will be collapsed into $x$, in particular $x$ itself. In the unlikely case that a data point is an element of multiple sets, we select one of the sets uniformly at random and remove the point from all but the selected set.

Finally, we assign the coarsened chunk $S$ new weights $w'$:

$$w'^*(x) = \sum_{y \in A_x} w(y).$$

In practice, to prevent having very large values for the individual weights, a user might wish to add a linear or logarithmic scaling to this weight assignment.

The clustering assignment induced by the Voronoi partition with respect to an $\varepsilon$-separated subset has the advantage that each point is within a distance of at most $2\varepsilon$ of the centroid, that is, the weighted mean of all points in a cluster. A Voronoi partition of $k$-means clustering does not necessarily have this property. Note that having this property is an advantage in the context of data approximation but can be a disadvantage when trying to identify spatially irregular clusters.

**D. Iterative Implementation**

The BiDViT algorithm repeats the procedure of data partitioning, chunk coarsening, and chunk collapsing with an increasing radius of interest, until the entire dataset collapses to a single data point. We refer to these iterations as BiDViT levels. If not specified, the radius of interest $\varepsilon_0$ in the initial level can be chosen such that a specific percentage of the data points of a randomly drawn sample from the entire dataset collapses. The increase of $\varepsilon$ between BiDViT levels is realized by multiplying $\varepsilon$ by a constant factor $\alpha$ that is specified by the user.

It is worth noting that, at each level, instead of proceeding with the identified representative data points (which result from coarsening the chunks), one can use the cluster centroids. Using these weighted cluster means at each level allows for more-accurate data coarsening and label assignment. By generating a tree, the algorithm keeps track of all BiDViT levels and which data points collapse into which centroids.

| original graph | 1 | 3 | 1 | 3 | $\omega(P) = 11$ |
| heuristic selection | 1 | 4 | $\omega(S_h) = 5$ |
| QUBO-based selection | 3 | 5 | $\omega(S_q) = 6$ |

**Figure 3:** Illustration of the advantage of the quantum method. The heuristic solver will first select the vertex with a weight of 4 and then the one with a weight of 1. We can see that the selected set is not optimal. The quantum method explores every possible combination and thus finding the best possible set $S_q$ is guaranteed.
at each particular level. The leaves of this tree represent the original data points, and the lower-level nodes represent the centroids of the respective BiDViT level; see Figure 5 for an example. Two leaves share a label with respect to a specific BiDViT level, say \( m \), if they have collapsed into centroids which, possibly after multiple iterations, have collapsed into the same centroid at the \( m \)-th level of the tree.

Note that, analogously to agglomerative (bottom-up) hierarchical clustering methods, BiDViT begins by assigning each point its own cluster and then proceeds to generate clustering assignments until every point has been assigned to that same cluster.

V. RELATION TO THE MAXIMUM WEIGHTED INDEPENDENT SET PROBLEM

The process of finding a maximal \( \varepsilon \)-separated set with maximum weight, described in Section III, is equivalent to solving the MWIS problem for the weighted graph \( G^\varepsilon = (P, E^\varepsilon, w_P) \).

Let \( G = (V, E, w) \) be a weighted graph. Recall that a set of vertices \( V_0 \subseteq V \) is called independent if no two of its vertices are adjacent. This corresponds to the separation constraint mentioned earlier, where two vertices are adjacent whenever they are less than \( \varepsilon \) apart. The MWIS problem can be posed as

\[
\begin{align*}
\text{maximize} & \quad \sum_{v \in V_0} w(v) \\
\text{subject to} & \quad \sum_{v \in V_0} \sum_{w \in V_0, w \neq v} 1 = 0.
\end{align*}
\]

The MWIS problem for a general weighted graph is NP-complete [1, 11, 17]. For specific graphs, polynomial-time algorithms have been developed [19, 28]. Note that the QUBO formulation of the MWIS problem in [1, 12] is related to the QUBO formulation for the coarsening step in our algorithm.

If all weights are positive, a maximum weighted independent set is maximal. Furthermore, a maximal independent set is a dominating set, that is, a subset \( S \) of \( V \) such that every \( v \in V \setminus S \) is adjacent to some \( w \in S \). This corresponds to our observation that every maximal \( \varepsilon \)-separated subset is \( \varepsilon \)-dense.

VI. COMPLEXITY ANALYSIS

We have divided the discussion on the order of complexity into two parts: data partitioning and data coarsening. As before, let \( n \) be the total number of \( d \)-dimensional data points in the original dataset and let \( \kappa \) denote the chunk cardinality. It should be noted that \( \kappa \ll n \).

1) Data partitioning: The order of complexity of the partitioning procedure is \( O(d n \log(n/\kappa)) \). This can be derived as follows. Note that there are at most \( \lceil \log_2(n/\kappa) \rceil \) partitioning stages and in the \( s \)-th stage we split \( 2^s - 1 \) chunks \( P_i \), where \( i = 1, \ldots, 2^s - 1 \). Let \( n_i \) denote the number of data points in chunk \( P_i \). Finding the dimension of maximum variance is \( O(d n_i) \) and determining the median of this dimension can be achieved in \( O(n_i) \) via the “median of medians” algorithm. Having computed the median, one can construct two chunks of equal size in linear time. Since \( \sum_{1 \leq i < 2^s - 1} n_i = n \), a partitioning step is \( O((dn + n)(\log_2(n/\kappa) + 1)) = O(d n \log(n/\kappa)) \).

It is worth noting that any division of a chunk is independent of the other chunks in the same stage; thus, this procedure can benefit from parallel (i.e., distributed) computing.

2) Data coarsening: The order of complexity for the collapsing process is \( O(d n \kappa) \). To see this, note that computing the neighbourhood matrix of a chunk is \( O(d \kappa^2) \) and that the heuristic selection procedure is \( O(\kappa^2) \). The number of chunks is bounded from above by \( \lceil n/\kappa \rceil \). This yields a complexity of \( O((n/\kappa)(d \kappa^2 + \kappa^2)) = O(d n \kappa) \). As data coarsening in each chunk is independent, the process of partitioning can also benefit from parallel processing. With \( \lceil n/\kappa \rceil \) parallel processors, the complexity reduces to \( O(d \kappa^2) \).

In sum, our analysis shows that the heuristic version of BiDViT has a computational complexity of \( O(d n \log(n/\kappa) + d n \kappa) \), and numerical experiments coincide with this result.

VII. RESULTS

In order to demonstrate the efficiency and robustness of the proposed approach to extreme clustering, we present the results of applying BiDViT to multiple datasets. The datasets used are the MNIST dataset of handwritten digits [22], a two-dimensional version of MNIST which we obtained by using t-SNE [27, 33], two synthetic grid datasets in two and in three dimensions, and a dataset called Covertype, which contains data on forests in northern Colorado and is publicly available at the University of California, Irvine machine learning repository [3]. The most important of these datasets are MNIST and Covertype, as they fit in the category of big data. An overview of the statistics on the datasets is provided by Table I. We provide results for both the extreme clustering domain and the non-extreme clustering domain.

A. Low-range clustering domain

Even though BiDViT has been specifically designed for the domain of extreme clustering, that is, large number of points
Figure 5: A dendrogram representing the output tree of BiDViT and the encoding of the clustering assignment. The original dataset is represented by the leaves $\ell_1, \ldots, \ell_9$. The leaves collapse into a single centroid after three BiDViT iterations. The first BiDViT iteration ("BiDViT level 1") results in four centroids $c_{1,1}, \ldots, c_{1,4}$. Each of these centroids corresponds to a cluster consisting of the nodes that collapsed into it. The centroids are the weighted average of the data points contained in their cluster. During the next iteration ("BiDViT level 2"), the algorithm merges the clusters of the centroids. For example, $c_{1,3}$ and $c_{1,4}$ are merged into a new centroid $c_{2,3}$ at the next level. At BiDViT level 0, each leaf is its own cluster, and at BiDViT level 3, all leaves are elements of the same, individual cluster.

Figure 6: Performance of BiDViT in the non-extreme clustering domain. The left-hand side figures show the original datasets (blue) with the cluster centroids (orange) determined by BiDViT. In the right-hand side figures, each colour corresponds to the assigned label.
$n$ and large number of clusters $k$, it yields desirable results for small $k$ as well. Recall that $k$ is not an input parameter. The results are obtained by running BiDViT iteratively and manually selecting a specific BiDViT level. Figure 6 shows the clustering assignment of BiDViT on the two-dimensional grid dataset (top figures) and on MNIST (bottom figures). In the grid dataset, every cluster has been identified correctly. In the MNIST dataset all clusters have been recognized, except one on the left-hand side that has been divided into two sub-clusters.

These datasets are quite specific in their geometry and there are datasets for which BiDViT will not accurately assign clusters. This is due to the fact that our algorithm is based on metric balls, and some datasets might not conform to such categorization. This is true for most clustering algorithms, as they are only able to recognize specific shapes.

B. Extreme clustering capability

To evaluate the performance of BiDViT on high-dimensional datasets in the extreme clustering range, we cannot simply analyze a scatter plot. However, there are well-known clustering metrics, for example, the Calinski–Harabasz score [4] and the Davies–Bouldin [7] score, which can be used to compare clustering algorithms. These clustering metrics are called internal evaluation schemes, that is, their values depend solely on the clustered data and they do not require the true labels for the dataset. This implies that they have to be viewed as heuristic methods in the sense that their optimal values do not guarantee optimal clusters but provide a reasonable measure of clustering quality. For a detailed analysis of the advantages and shortcomings of specific internal clustering measures, we refer the reader to [14, 24].

1) The Calinski–Harabasz score. Let $C_1, \ldots, C_{n_c}$ denote a total of $n_c$ detected clusters within a dataset $X$ with $n$ data points. The Calinski–Harabasz score $S_{CH}$ of a clustering is defined as a weighted ratio of the inter-cluster squared deviations, which measures how well the clusters are separated, and the sum of the intra-cluster squared deviations, which measures the compactness of the clusters. More precisely, the Calinski–Harabasz score is given by the expression

\[ S_{CH}(C_1, \ldots, C_{n_c}) = \frac{(n-c-1) \sum_{k=1}^{n_c} |C_k||c_k-c|_2^2}{\sum_{k=1}^{n_c} \sum_{x \in C_k} \|x-c_k\|_2^2}, \]

where $c_k$ is the cluster centroid of cluster $C_k$ for $k = 1, \ldots, n_c$ and $c$ is the mean of the set of cluster centroids. High values of $S_{CH}$ are indicative of a high quality of clustering.

| Name       | Description         | Cardinality (in thousands) | Dimension |
|------------|---------------------|-----------------------------|-----------|
| MNIST      | handwritten images  | 60                          | 784       |
| MNIST-2D   | t-SNE of the above  | 60                          | 2         |
| Covertype  | forest data         | 581                         | 54        |
| grid-2D    | synthetically generated | 100                         | 2         |
| grid-3D    | synthetically generated | 1000                        | 3         |

*Table 1: Dataset Statistics*

Figure 7: Quality of solutions obtained with the heuristic BiDViT algorithm ($\kappa = 1000, \alpha = 1.2$) and Mini Batch $k$-means clustering (batch size 50, maximum iterations 1000, tolerance: 0.001, number of initializations: 1) measured with respect to clustering metrics. A higher value of the Calinski–Harabasz score indicates better clustering. The opposite is the case for the Davies–Bouldin score.
2) Davies–Bouldin score. The Davies–Bouldin score $S_{DB}$ is a measure of the average worst case of the pairwise ratios of the intra-cluster deviation and the inter-cluster distance. The score is obtained by computing

$$S_{DB}(C_1, \ldots, C_n) = \frac{1}{n_c} \sum_{k=1}^{n_c} \max_{j \neq k} \frac{S_k + S_j}{\|c_k - c_j\|_2},$$

where $S_i = |C_i|^{-1} \sum_{x \in C_i} \|x - c_i\|$. Low values of $S_{DB}$ are indicative of accurate clustering. Note that, whereas the Calinski–Harabasz score is based on squared distances, the Davies–Bouldin score uses non-squared distances.

Figure 8 shows $S_{CH}$ and $S_{DB}$ of clustering assignments obtained with BiDViT and Mini Batch $k$-means clustering [32] for different values of $k$ on the Covertype dataset (top figures) and the MNIST dataset (bottom figures). Due to their high computational complexity with respect to the number of clusters $k$, many common clustering algorithms could not be applied to the Covertype dataset. Remarkably, $S_{CH}$ values for both datasets are quite similar. This indicates that the clusters assignments generated by BiDViT are of comparable quality even though the runtime of our algorithm is significantly shorter. With respect to $S_{DB}$, our algorithm outperforms most other algorithms for lower values of $k$, and is comparable for large values. One explanation for the slightly weaker performance of BiDViT with respect to $S_{CH}$ is that BiDViT aims to minimize the Euclidean distances, whereas $S_{CH}$ rewards clustering methods that minimize squared distances. Similarly, this explains the advantage of our algorithm when comparing $S_{DB}$.

We obtain similar results when comparing the Silhouette scores [31] on subsets of the datasets. Due to the limited computational resources available to us, we did not compute the values of the Silhouette score on the entire dataset.

C. Runtime Comparison

The purpose of designing BiDViT has been to develop a fast extreme clustering algorithm which is able to harness the power of both multiprocessing and quantum annealing. We observe that even the single-core heuristic version of our algorithm outperforms common clustering methods in the extreme clustering domain with respect to the total runtime.

An application of classical $k$-means clustering is practically impossible in the extreme clustering domain, due to the high computational complexity with respect to the number of clusters and the number of data points. Ordinary methods, such as Mini Batch $k$-means clustering, have improved computational complexity with respect to the number of data points $n$ but not with respect to a high number of clusters $k$.

Agglomerative clustering algorithms, such as Ward [35] clustering, are capable of handling large values of $k$; however, they typically have at least quadratic dependence on $n$ and thus require enormous computational resources.

Figure 8 shows the runtime required by different clustering algorithms on the Covertype dataset. For the implementation of methods other than BiDViT, we used the publicly available sklearn.clustering module for Python. To generate the plots, we ran the entire BiDViT procedure once and then applied the classical algorithms for the same values of $k$.

The results suggest that, in the extreme clustering domain, the runtime of BiDViT is an order of magnitude faster than that of the agglomerative methods and multiple orders of magnitude faster than that of $k$-means and Mini Batch $k$-means clustering. Note that we had to restrict the cardinality of the dataset to 20,000 points to obtain results for methods other than BiDViT. For larger datasets, these other algorithms had either enormous runtimes or generated memory errors. Our algorithm is capable of handling the entire dataset with its 581,000 points.
Finally, we compared the runtime of BiDViT to PERCH ("Purity Enhancing Rotations for Cluster Hierarchies"), an online hierarchical algorithm for extreme clustering [18] (publicly available at the GitHub repository referenced in the paper). To our knowledge, it is the only other algorithm designed to solve extreme clustering problems. To compare the results of PERCH and BiDViT, we restricted both algorithms to using a single core. Table II shows that BiDViT performs an order of magnitude faster than PERCH. However, we would like to point out that BiDViT and PERCH solve somewhat different problems; whereas BiDViT aims to gradually coarsen the dataset by finding \( \varepsilon \)-separated, \( \varepsilon \)-dense subsets, PERCH has been developed to maximize the dendrogram purity, which is a measure of the consistency of the clustering tree (see [18] for a technical definition). The outputs of both algorithms differ. The clustering tree generated by PERCH is binary and therefore enormous in size, which allows a very fine incremental distinction between clusters. In contrast, the tree generated by BiDViT is more compact, due to the fact that multiple data points can collapse into the same representative point. When comparing dendrogram purities, we expect PERCH to outperform BiDViT, and when comparing Davies–Bouldin scores at a given level, we expect that BiDViT outperforms PERCH. We did not conduct an experiment to test these expectations, as the dendrogram purity is an external evaluation scheme, that is, it requires a clustering assignment to use for comparison, which is not available in unsupervised machine learning.

D. Results for the quantum version of BiDViT

Quantum annealing is a process that can be used to find the optimal solution to optimization problems that can be encoded in QUBO form or, using physics terminology, by an Ising spin model [9]. This is the case for many NP-hard problems and has been widely applied [26].

We tested an early prototype of our algorithm on a D-Wave 2000Q quantum annealer. This machine has a total of 2048 qubits and 5600 couplers. According to D-Wave Systems, to make this possible, the computer uses 128,000 Josephson junctions and was by far the most complex superconducting integrated circuit ever built at the time of its introduction in January of 2017 [6].

In our experiments, we observed an even higher-quality solution and a significant speed-up, when compared to our heuristic algorithm and common clustering methods. Both of these observations are based on Figure 9a and Figure 9b. However, we wish to point out that the heuristic version of BiDViT and the common clustering algorithms were executed on a classical device that has a limited computational capacity, whereas the D-Wave 2000Q is a highly specialized device. Running these algorithms on a high-performance computer might lead to an equivalence in the degree of speed-up.

**Figure 9:** Runtime and quality results for the quantum version of BiDViT obtained using a D-Wave 2000Q quantum annealer.

| Algorithm specified parameters | Runtime on Dataset (seconds) |
|--------------------------------|-----------------------------|
| PERCH-C \( L = \infty \)      | 1616.45 ± 20.37 1588.10 ± 41.46 |
| PERCH-C \( L = 100,000 \)     | 1540.18 ± 18.34 1487.14 ± 30.87 |
| PERCH-C \( L = 50,000 \)      | 1232.53 ± 53.61 1280.30 ± 15.03 |
| PERCH-C \( L = 10,000 \)      | 928.82 ± 47.00  – |
| BiDViT (heuristic) \( \kappa = 2000, \alpha = 1.1 \) | 301.36 ± 10.01 152.50 ± 0.86 |
| BiDViT (heuristic) \( \kappa = 2000, \alpha = 1.2 \) | 125.56 ± 1.37 99.20 ± 0.83 |
| BiDViT (heuristic) \( \kappa = 1000, \alpha = 1.2 \) | 77.66 ± 0.48 80.25 ± 0.55 |
| BiDViT (heuristic) \( \kappa = 500, \alpha = 1.2 \) | 56.26 ± 0.62 75.22 ± 0.95 |

Table II: Runtime of extreme clustering algorithms in seconds (mean and standard deviation over five runs). PERCH is run in “collapse mode” (see [18] for an explanation). The parameter \( L \) determines the maximum number of leaves. At BiDViT level 1, the initial radius \( \varepsilon_0 \) was chosen automatically such that a certain percentage of data points collapsed. For the Covertype dataset the algorithm selected \( \varepsilon_0 = 30 \), and for the grid dataset BiDViT selected \( \varepsilon_0 = 0.5 \). The parameter \( \kappa \) denotes the chunk cardinality. The parameter \( \alpha \) denotes the factor by which we increase the radius of interest between two successive BiDViT levels.
VIII. CONCLUSION

We have developed an efficient algorithm capable of performing extreme clustering. Our complexity analysis and numerical experiments show that if the dataset cardinality \( n \) and the desired number of clusters \( k \) are both large, the runtime of BiDVIT is at least an order of magnitude faster than that of classical algorithms, while yielding a solution of comparable quality. With advances in quantum annealing hardware, one can expect further speed-ups.

The fact that our coarsening method is based on identifying an \( \varepsilon \)-dense, \( \varepsilon \)-separated subset—a novel approach to clustering problems not limited to extreme clustering problems—is interesting in its own right, independent of BiDVIT. Within our proof of Theorem 2, we have identified a domain for the radius of interest such that, under a separability assumption, every solution to Problem (1) yields an optimal clustering assignment, justifying further investigation of this approach.

ACKNOWLEDGEMENT

We thank Austin Wallace, Saeid Allahdadian, and Daniel Crawford for contributing to an earlier version of the algorithm. We are especially grateful to Marko Bucyk for carefully reviewing and editing the manuscript and providing valuable feedback. Furthermore, we thank Pooja Pandey for a helpful discussion on the maximum weighted independent set problem, Maliheh Aramon for insights on efficient QUBO formulations of the MWIS problem, and Brad Woods for pointing out the similarities between minimal \( \varepsilon \)-dense sets and the dominating set problem. Partial funding for this work was provided by the Mitacs Accelerate internship initiative.

REFERENCES

[1] A. A. Abbott, C. S. Calude, M. J. Dinneen, and R. Hua, A Hybrid Quantum-Classical Paradigm to Mitigate Embedding Costs in Quantum Annealing, arXiv:1803.04340 (2018).
[2] S. Balaji, V. Swaminathan, and K. Kannan, Approximating Maximum Weighted Independent Set Using Vertex Support, International Journal of Computational and Mathematical Sciences 3 (2009), no. 8, 406–411.
[3] J. A. Blackard, UCI Machine Learning Repository”, University of California, Irvine, School of Information and Computer Sciences, 2017.
[4] T. Caliński and J. Harabasz, A Dendrite Method for Cluster Analysis, Commun. Stat. Theory Methods 3 (1974), no. 1, 1–27.
[5] T. Coulhon and L. Saloff-Coste, Variétés riemanniennes isométriques à l’infini, Revista Matemática Iberoamericana 11 (1995), no. 3, 687–726. MR1363211
[6] D-Wave Systems Inc., The D-Wave 2000Q Quantum Computer: Technology Overview, 2017. [Online; accessed: 2019-02-13].
[7] D. L. Davies and D. W. Bouldin, A Cluster Separation Measure, IEEE Transactions on Pattern Analysis and Machine Intelligence 2 (1979), 224–227.
[8] C. M. de Vries, L. de Vine, S. Geva, and R. Nayak, Parallel Streaming Signature EM-tree: A Clustering Algorithm for Web Scale Applications, Proceedings of the 24th International Conference on World Wide Web, 2015, pp. 216–226.
[9] A. B. Finnila, M. A. Gomez, C. Sebenik, C. Stenson, and J. D. Doll, Quantum Annealing: A New Method for Minimizing Multidimensional Functions, Chemical Physics Letters 219 (1994), no. 5, 343–348.
[10] Fujitsu Ltd., Digital Annealer Introduction: Fujitsu Quantum-inspired Computing Digital Annealer, 2018. [Online; accessed: 2019-02-13].
[11] M. R. Garey and D. S. Johnson, Computers and Intractability, Vol. 29, W. H. Freeman & Co., New York, NY, USA, 2002.
[12] M. Hernandez, A. Zarifbayan, M. Aramon, and M. Naghibi, A Novel Graph-Based Approach for Determining Molecular Similarity, arXiv:1601.06693 (2016).
AN ALTERNATIVE COARSENING METHOD

In specific situations, a user might not want the approximating set to be \( \varepsilon \)-separated but instead might be interested in finding an \( \varepsilon \)-dense subset with a minimum number of elements, or, more generally, the minimum cost for some cost function \( c: X \rightarrow \mathbb{N} \). Finding such a set can be realized in a very similar way to the quantum method of BiDiViT. In fact, the only modifications needed would be to Section IV-B, where we introduce chunk coarsening.

Let \( P = \{ x(1), \ldots, x(n) \} \) and let \( N(^e) \) and \( s_i \) for \( i = 1, \ldots, n \) be defined as in Section IV-B. Analogously to the weight vector \( w \), we define a cost vector \( c \) by \( c_i = c(x(i)) \) for each \( x(i) \in P \). The problem of finding an \( \varepsilon \)-dense subset \( S \subseteq P \) of minimum cost can then be expressed as follows:

\[
\min_{s \in \{0,1\}^n} \sum_{i=1}^n s_ic_i
\]
\[\text{subject to } \sum_{j=1}^n N_{ij}(^e)s_j \geq 1, \quad i = 1, \ldots, n.
\] (6)

The constraints in Problem (6) enforce that for each solution (corresponding to a subset), every point in \( P \) is represented (at least one of the points from the selected subset. The subset will not necessarily be \( \varepsilon \)-separated, but it will be \( \varepsilon \)-dense.

In the same way that finding an \( \varepsilon \)-separated subset of maximum weight corresponds to the MWIS problem, finding an \( \varepsilon \)-dense subset of minimum cost corresponds to the minimum weighted dominating set (MWDS) problem, which is equivalent to a weighted version of the minimal set covering (MSC) problem.

Consider a set \( U \) and subsets \( S_j \subseteq U \) and \( j \in J \), where \( J \) is some set of indices, such that \( U = \bigcup_{j \in J} S_j \). The MSC problem then consists of finding a subset \( J_0 \subseteq J \) such that the property \( U \subseteq \bigcup_{j \in J_0} S_j \) is satisfied, and \( J_0 \) is of minimum cardinality with respect to this property. For example, if \( U = \{a,b,c,d,e\}, S_1 = \{a,c\}, S_2 = \{a,d\}, \) and \( S_3 = \{b,d,e\} \), then the solution to the MSC problem is given by \( J_0 = \{1,3\} \), as none of the subsets cover \( U \) but the union \( S_1 \cup S_3 \) does.

The general MSC problem is known to be NP-hard [17,26].

By defining \( S_j = B(x(j),^e) \cap P \) for \( j = 1, \ldots, n \) in the above setting, one can see that we have solved a weighted version of the MSC problem.

To transform Problem (6) into a QUBO problem, we convert the inequality constraints to equality constraints by adding integer slack variables. Note that the \( i \)-th constraint is satisfied if and only if there exists some \( \xi_i \in \mathbb{N}_0 \) such that

\[
\left( \sum_{j=1}^n N_{ij}(^e)s_j \right) - 1 = \xi_i.
\]

In fact, given that \( s \in \{0,1\}^n \), we can see that such a \( \xi_i \) must satisfy the bounds

\[
0 \leq \xi_i \leq \left( \sum_{j=1}^n N_{ij}(^e) \right) - 1.
\] (7)

Therefore, by dualizing the equality constraints, Problem (6) can be expressed as a quadratic unconstrained integer optimization problem:

\[
\min_{s \in \{0,1\}^n} \sum_{i=1}^n s_is_i + \lambda \sum_{i=1}^n \left( \sum_{j=1}^n N_{ij}(^e)s_j \right) - 1 - \xi_i.
\] (8)

We will now describe how substituting a binary encoding for each of the \( \xi_i \), \( i = 1, \ldots, n \) in Problem (8) yields the desired QUBO formulation. For each \( i = 1, \ldots, n \), the \( (N(^e)_{ij}) \) possible states of \( \xi_i \) can be encoded by \( [m_i] + 1 \) binary variables \( b^{(i)}_0, \ldots, b^{(i)}_{m_i} \), where \( m_i = \log_2(N(^e)_{ij}) \). The encoding has the form

\[
\xi_i = \sum_{k=0}^{m_i} b^{(i)}_k, \quad \text{for } i = 1, \ldots, n,
\]

where \( \gamma^{(i)}_k \in \mathbb{N} \) are fixed coefficients that depend solely on the bounds (7). If we were to select \( \gamma^{(i)}_k = 2^k \) for \( k = 0, \ldots, [m_i] \), then, if \( m_i \notin \mathbb{N} \), \( \xi_i \) could assume states that do not satisfy these bounds. We can avoid this situation by manipulating the coefficient \( \gamma^{(i)}_k \) of the final bit \( b^{(i)}_{m_i} \) such that \( \sum_{k=0}^{[m_i]} 2^k + \gamma^{(i)}_{[m_i]} = (N(^e)_{ij}) - 1 \). This may lead to a situation where there are multiple valid encodings for the same integer but it will always hold that

\[
0 \leq \sum_{k=0}^{[m_i]} b^{(i)}_k \leq \left( \sum_{j=1}^n N_{ij}(^e) \right) - 1,
\]

where \( \gamma^{(i)}_k = 2^k \) for \( k < [m_i] \). Substituting the binary encoding into Problem (8) yields the following QUBO formulation:

\[
\min_{b^{(i)} \in \{0,1\}^{[m_i]+1}} \sum_{i=1}^n s_is_i + \lambda \left( \sum_{j=1}^n N_{ij}(^e)s_j \right) - 1 - \sum_{k=0}^{[m_i]} b^{(i)}_k \gamma^{(i)}_k.
\]

One can show that the solution set of this QUBO problem is equivalent to the one for Problem (6) for \( \lambda > n|c| \). We have not investigated whether this bound is sharp. Note that our QUBO formulation is similar to the one described in [26], but uses a different encoding.

The number of binary variables in the QUBO formulation of this problem depends on the binary encoding of \( \xi \). If the vertex degree in \( G^\varepsilon \) is uniformly bounded from above by a constant \( \eta > 0 \), then each \( \xi_i \) can be encoded with fewer than \( \lfloor \log_2(\eta) \rfloor \) binary variables. Therefore, the number of variables in the QUBO polynomial will be at most \( n(1 + \lfloor \log_2(n) \rfloor) \). In the worst case of there being a vertex that is a neighbour of every other vertex, the polynomial would still comprise fewer than \( n(1 + \lfloor \log_2(n) \rfloor) \) variables.