A Quantum Annealing-Based Approach to Extreme Clustering

Tim Jaschek[1,2], Marko Bucyk[1], and Jaspreet S. Oberoi[1,3]

[1] 1QB Information Technologies (1QBit), Vancouver, BC, Canada
[2] Dept. of Mathematics, University of British Columbia, Vancouver, BC, Canada
[3] School of Engineering Science, Simon Fraser University, Burnaby, BC, Canada

Abstract. There is a growing need for algorithms and techniques capable of organizing big data in an accurate and efficient manner. Clustering, or grouping dataset elements based on similarity, can be computationally expensive, especially when employed on massive datasets to divide them into a relatively large number of groups. The task of clustering what can amount to millions (billions) of data points into thousands (millions) of clusters is referred to as *extreme clustering*. We have devised a distributed method that can be employed to efficiently solve extreme clustering problems using a quantum annealer.

Keywords: extreme clustering · distributed computing · quantum computing · maximum weighted independent set · unsupervised learning

1 Introduction

Traditionally, clustering approaches have been developed and customized for tasks where the resultant number of clusters is not particularly high. In such cases, algorithms such as k-means, BIRCH [1], and spectral clustering produce high-quality solutions in a reasonably short time. This is because these traditional algorithms scale well with respect to the number of data points n. However, in most cases, the computational complexity of these algorithms, in terms of the number of clusters k, is either exponential or higher-order polynomial. Another common problem is that some of the algorithms require vast amounts of memory.

The demand for clustering algorithms is continually increasing to larger values of k. Present-day examples involve deciphering the content of billions of web pages by grouping them into millions of topics (or "labels") [2, 3], identifying duplicates among billions of images using nearest-neighbour detection [4, 5], and clustering hundreds of billions of satellite images into a few billion categories [6]. This domain of clustering, where n and k are both substantially large, has been referred to as *extreme clustering* [7]. Although there is great value in perfecting this type of clustering, very little effort towards this end has been made by the

* T. Jaschek and J. S. Oberoi have contributed equally to the manuscript. Address correspondence to: tim.jaschek@1qbit.com

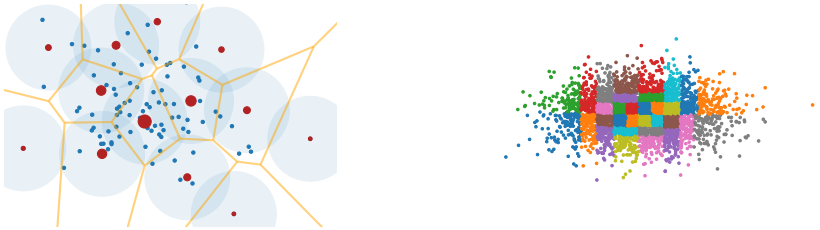
machine learning community. Our algorithm is, in fact, such an effort. Its output is a *clustering tree*, which can be used to generated multiple clustering assignments (or "levels") with varying degrees of accuracy. Generating such a tree is not uncommon for clustering algorithms. Consider, for example, hierarchical clustering algorithms which generate binary clustering trees. Clustering trees are useful tools for dealing with real-world data visualization problems. Our algorithm, the *Big Data Visualization Tool*, or *BiDViT*, provides this functionality.

BiDViT employs a novel approach to clustering problems, which is based on the maximum weighted independent set (MWIS) problem in a graph induced by the original dataset and a parameter we call the *radius of interest*. The MWIS problem can be transformed into a quadratic unconstrained binary optimization (QUBO) problem, the formulation accepted by a quantum annealer. An alternative way to address the underlying problem is to use a heuristic algorithm to approximate solutions to the MWIS problem. Quantum annealing and simulated annealing have previously been applied in clustering. Consider, for example, [8] and [9], where quantum annealing is used to solve clustering problems similar to the k-means clustering problem, and [10], which studies the application of quantum annealing for probabilistic clustering models such as Gaussian mixture models. However, the mentioned approaches are not capable of addressing problems in the extreme clustering domain.

We prove that, under a separability assumption on the ground truth clustering of the original dataset, our method yields the optimum clustering assignment. We provide results on runtime and solution quality for both versions of our algorithm, with respect to external evaluation schemes such as the Calinski–Harabasz and the Davies–Bouldin scores. Our results suggest that BiDViT yields clustering assignments of a quality comparable to that of assignments generated by common clustering algorithms, yet does so a full order of magnitude faster.

2 The Coarsening Method

Our algorithm is based on a combinatorial clustering method, which we call *coarsening*. The key idea behind coarsening is to approximate a set $X \subset \mathbb{R}^d$ by a subset $S \subseteq X$ such that, for any point $x \in X$, there exists $y \in S$ such that $\|x-y\|_2 < \varepsilon$, for some parameter $\varepsilon > 0$. In this case, we say that S is ε-*dense* in X and call ε the *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, equipped with a metric $d : X \times X \to [0, \infty)$. Finding an arbitrary ε-dense subset of X does not necessarily provide us with a helpful approximation. For example, X itself is always ε-dense in X. However, enforcing the additional constraint that any two points in the subset S must be separated by a distance of at least ε yields more-interesting approximations, which often lead to a reduction in the number of data points (one of our primary objectives). We call such a set ε-*separated*.



(a) A maximal ε-separated subset (red dots) of a dataset (red dots and blue dots). The circles have a radius equal to the radius of interest ε. The weights of the red points are updated according to the number of blue points within a distance of ε. The yellow borders are a Voronoi partition of the dataset that indicates the clustering assignment.

(b) Data partitioning of a dataset along the axes of maximum variance. In this example, there are $s = 5$ partitioning steps, resulting in $2^5 = 32$ chunks.

Fig. 1: Visualization of (a) chunk collapsing and (b) data partitioning.

Fig. 1a shows an example of a point cloud and an ε-dense, ε-separated subset. The theorem that follows shows that a maximal ε-separated set S of X is necessarily ε-dense in X.

Theorem 1. *Let S be a maximal ε-separated subset of X in the sense of set inclusion. Then S is ε-dense in X. Furthermore, S is a minimal ε-dense subset.*

Proof. Let S be a maximal ε-separated subset of X and assume, in contradiction, that S is not ε-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 ε-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 ε-separated, $d(x, y) \geq \varepsilon$ for any $y \in S$ and, thus, $S \setminus \{x\}$ is not ε-dense in X. □

Note that a maximal ε-separated subset does not refer to an ε-separated subset with fewer than or equally as many elements as all other ε-separated subsets but, rather, to an ε-separated subset that is no longer ε-separated when a single data point is added. Contrary to Thm. 1, a minimal ε-dense subset does not need to be ε-separated. Consider the set $X = \{1, 2, 3, 4\} \subset \mathbb{R}$, and let d be the Euclidean distance on $\mathbb{R}$. 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 *weight function* $w : X \to \mathbb{R}_+$. We call $w_i = w(x^{(i)})$ the *weight* of $x^{(i)}$ and gather all weights in a *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 $S \subseteq X$ is given by $\omega(S) = \sum_{x \in S} w(x)$. We are interested in solving the optimization problem

$$\underset{S \subseteq X}{\text{maximize}} \ \omega(S) \quad \text{subject to} \quad S \text{ is } \varepsilon\text{-separated.} \tag{P1}$$

If we impose unit weights, the solution set to this optimization problem will consist of the maximal ε-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 (P1) will always be a maximal ε-separated subset and, therefore, by Thm. 1, ε-dense. In Sec. 3.6, we show that this problem is equivalent to solving an MWIS problem for a weighted graph $G^\varepsilon(X, E^\varepsilon, w)$, depending solely on the dataset X, the Euclidean metric d, and the radius of interest ε. Thus, the computational task of finding a maximal ε-separated subset of maximum weight is NP-hard [11,12]. Note that an ε-separated subset S gives rise to a *clustering assignment* $\mathcal{C} = \{C_x\}_{x\in S}$. This assignment is given by

$$C_x = \{y \in X : d(x, y) \leq d(x', y) \text{ for all } x' \in S\}. \tag{1}$$

More details on the clustering assignment are provided in Sec. 3. A common assumption in the clustering literature is *separability*—not to be mistaken with ε-separability—of the dataset with respect to a clustering $\mathcal{C}$. The dataset X is called *separable* with respect to a clustering $\mathcal{C} = \{C_1, \dots C_k\}$ if

$$\max_{\substack{x,y\in C_i \\ 1\leq i\leq k}} d(x, y) < \min_{\substack{x\in C_i, y\in C_j \\ 1\leq i\neq j\leq k}} d(x, y), \tag{2}$$

that is, if the maximum *intra-cluster* distances are strictly smaller than the minimum *inter-cluster* distances. The following theorem shows that, if ε is chosen correctly, our coarsening method yields the clustering assignment $\mathcal{C}$. For simplicity, we denote the left- and right-hand sides in (2) by l and r, respectively.

Theorem 2. *Let X be separable with respect to a clustering $\mathcal{C} = \{C_1, \dots C_k\}$. Then, for any $\varepsilon \in (l, r]$, the coarsening yields the correct clustering assignment.*

Proof. By the separability assumption, the above interval is non-empty. One can see that, for any admissible choice of ε, any two points from different clusters are ε-separated. Indeed, for $x \in C$ and $y \in C'$, it holds that $d(x, y) \geq r \geq \varepsilon$. Furthermore, if a point x in a cluster C is selected, then no other point y in the same cluster can be selected, as $d(x, y) \leq l < \varepsilon$. Therefore, every solution $S \subseteq X$ to (P1) is a union of exactly one point from each cluster. Using the separability of X with respect to $\mathcal{C}$, we can see that the clustering assignment induced by (1) is coincident with $\mathcal{C}$. □

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

We have developed two methods, which we refer to as the *heuristic method* and the *quantum method*, to address the NP-hard task of solving (P1). The heuristic method loosens the condition of having a maximum weight; it can be seen as a greedy approach to (P1). In contrast, the quantum method explores all different maximal ε-separated subsets simultaneously, yielding one that has maximum weight. The quantum method is based on the formulation of a QUBO problem, which can be solved efficiently using a quantum annealer like the D-Wave 2000Q [13] or a digital annealer like the one developed by Fujitsu [14].

Both our methods can be used to construct ε-nets of metric measure spaces (M, d, μ). 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 [15, 16].

3 The Algorithm

Let $X = \{x^{(1)}, \ldots, x^{(n)}\} \subset \mathbb{R}^d$ denote a dataset of n d-dimensional data points. Note that, mathematically speaking, a dataset is not a set but rather a *multiset*, that is, repetitions are allowed. Our algorithm consists of two parts: *data partitioning* and *data coarsening*, the latter of which can be further subdivided into *chunk coarsening* and *chunk collapsing*.

3.1 Data Partitioning

In general, the computational complexity of distance-based clustering methods is proportional to the square of the dataset cardinality, as all pairwise distances need to be computed. We overcome this bottleneck by employing a *divide-and-conquer* approach. 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.

A *partition* $\mathcal{P}$ of X is a collection of non-empty disjoint sets $P_1, \ldots, P_k \subset X$ such that $X = \bigcup_{P \in \mathcal{P}} P$. Elements of partitions are typically referred to as blocks, parts, or cells; however, we refer to them as *chunks* throughout this paper. The partitioning 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 as it makes it possible for a fixed-sized processor to process the entire set of chunks. The number of points a user desires to have in a chunk is referred to as *maximum chunk cardinality* and denoted by κ. A user should make the determination of the value of κ by taking into account the number of available processors, their data handling capacity, or, in the case of a quantum annealer, the number of fully connected qubits.

To break the data into chunks, we pick the index of an axis of maximum variance uniformly at random among the elements of the set $\arg\max_{1 \le j \le d} \{\sigma_j^2\}$, where $\sigma_j^2 = \sum_i (x_j^{(i)} - \mu_j)^2/n$ and $\mu_j = \sum_i x_j^{(i)}/n$ are defined as usual. We then split the dataset along the selected axis, say ℓ, at the median m of $\{x_\ell^{(1)}, \ldots, x_\ell^{(n)}\}$ in such a way as to obtain two *data chunks* P_1 and P_2 whose cardinalities differ by at most one (in the case where n is odd) and which satisfy $P_1 \subseteq \{x \in X : x_\ell \le m\}$ and $P_2 \subseteq \{x \in X : x_\ell \ge m\}$. We cannot simply assign $P_1 = \{x \in X : x_\ell \le m\}$ and $P_2 = X \setminus P_1$, as these sets might differ drastically in cardinality. For example, when $x_\ell^{(1)} = \ldots = x_\ell^{(n)}$, 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, until the number of data points in the chunk to be divided is less than or equal to

the maximum chunk cardinality κ, yielding a binary tree of data chunks. After s iterations, this leaves us with 2^s chunks $P_k^{(s)}$ such that $X = \bigcup_{1\leq k\leq 2^s} P_k^{(s)}$, where the union is disjoint. Fig. 1b provides a visualization.

3.2 Chunk Coarsening

The goal of a data coarsening step is, for each chunk, to find *representative* data points such that the resultant representative point cloud, having fewer data points, can replace the original point cloud, while maintaining the original data distribution as accurately as possible. 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.

Now, let $P = \{x^{(1)}, \ldots, x^{(n)}\}$ be a chunk and $\varepsilon > 0$ be the radius of interest, which encodes the extent of similarity of two data points. 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 ε induces a weighted graph $G^\varepsilon = (P, E^\varepsilon, w_P)$, where P is the vertex set, the edge set E^ε is given by the relation $\sim_\varepsilon$ defined by

$$x^{(i)} \sim_\varepsilon x^{(j)} \quad \Leftrightarrow \quad d(x^{(i)}, x^{(j)}) < \varepsilon, \tag{3}$$

and the weight function $w_P : P \to \mathbb{R}_+$ is the restriction of w to 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 $x^{(j)}$ is an element of the open metric disk (in the 2D case) or the open metric ball (in the case of arbitrary dimensions) of radius ε 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 ε-dense in P.

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* $N^{(\varepsilon)}$ of the graph $G^\varepsilon = (P, E^\varepsilon, w_P)$ by $N_{ij}^{(\varepsilon)} = 1$ if $x^{(i)} \sim_\varepsilon x^{(j)}$, and $N_{ij}^{(\varepsilon)} = 0$ otherwise. Problem (P1) can then be posed as a quadratically constrained quadratic program (QCQP) given by

$$\underset{s\in\{0,1\}^n}{\text{maximize}} \sum_{i=1}^{n} s_i w_i \quad \text{subject to} \quad \sum_{i=1}^{n}\sum_{j>i} s_i N_{ij}^{(\varepsilon)} s_j = 0. \tag{P2}$$

Here, the inner summation of the constraint does not need to run over all indices, due to the symmetry of $N^{(\varepsilon)}$. The matrix form of (P2) is given by maximizing $s^T w$ subject to the constraint $s^T \overline{N}^{(\varepsilon)} s = 0$, where $\overline{N}^{(\varepsilon)}$ is the upper triangular matrix of $N^{(\varepsilon)}$ with all zeroes along the diagonal. There exists a wide range of software solvers for QCQPs but, as we will see in Sec. 3.6, (P2) 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 present two methods we have devised to address (P2).

The Heuristic Method We wish to emphasize that the heuristic method does not provide us with a solution to (P2). Rather, the aim of this method is to obtain an ε-separated subset S with a high—but not necessarily the maximum—weight $\omega(S)$. The seeking of approximate solutions to the MWIS problem is a well-studied subject [17–19]. We would like to draw particular attention to [19], which provides insight as to the approximation rates that have been achieved by the community. Taking this into consideration, we have designed our heuristic focusing on simplicity and low computational complexity.

The heuristic algorithm begins by selecting one of the data points $x^{(\ell_1)}$ uniformly at random among all maximizers of the weight function w, that is, it selects an index such that $\ell_1 \in \arg\max_{i=1,\dots,n} \{w_i\}$. The choice of $x^{(\ell_1)}$ has the consequence that the data points $x^{(i)}$ with $x^{(i)} \sim_\varepsilon x^{(\ell_1)}$ can no longer be elements of the set S^*. Hence, the next index ℓ_2 must be selected from the set $\{1 \le i \le n : x^{(i)} \nsim_\varepsilon x^{(\ell_1)}\}$. Having chosen k elements $\ell_1, \dots, \ell_k$, we select the $(k+1)$-th element uniformly at random such that $\ell_{k+1} \in \arg\max_{i \in I_k} \{w_i\}$, where $I_k = \{1 \le i \le n : x^{(i)} \nsim_\varepsilon x^{(\ell)},\ \ell = \ell_1, \dots, \ell_k\}$. This procedure can be repeated iteratively until $I_k = \emptyset$. In each step, we add the data point that 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; selecting such a locally suboptimal point might lead to situations where we could possibly select higher weights in subsequent steps, resulting in a higher cumulative weight. The supplementary material contains a pseudocode form of our heuristic method.

The Quantum Method In contrast to the heuristic method, the QUBO approach provides an actual (i.e., non-approximate) solution to (P2). 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 λ, the solution set of (P2) is equivalent to that of the QUBO problem

$$\underset{s \in \{0,1\}^n}{\text{maximize}} \ \sum_{i=1}^{n} s_i w_i - \lambda \sum_{i=1}^{n} \sum_{j>i} s_i N_{ij}^{(\varepsilon)} s_j. \tag{P3}$$

One can show that, for $\lambda > \max_{i=1,\dots n} w_i$ every solution to (P3) satisfies the separation constraint [20, Thm. 1]. Instead, we use individual penalty terms λ_{ij}, as this may lead to a QUBO problem with much smaller coefficients, which results in improved performance when solving the problem using a quantum annealer. Expressing (P3) as a minimization, instead of a maximization, problem and using matrix notation yields the problem

$$\underset{s \in \{0,1\}^n}{\text{minimize}} \ s^T Q s, \tag{P4}$$

where $Q_{ij} = -w_i$ if $i = j$, $Q_{ij} = \lambda_{ij}$ if $N_{ij}^{(\varepsilon)} = 1$ and $i < j$, and $Q_{ij} = 0$ otherwise. It is worth noting that the number of variables in this QUBO formulation is equal to the number of data points to be coarsened.

The following theorem show that (P2) is equivalent to (P4) for a suitable choice of λ_{ij}, for $1 \leq i < j \leq n$. We refer the interested reader to the supplementary material for a proof.

Theorem 3. *Let* $\lambda_{ij} > \max\{w_i, w_j\}$ *for all* $1 \leq i < j \leq n$. *Then, for any solution* $s \in \{0,1\}^n$ *to* (P4), *the corresponding set* $S \subseteq X$ *is* ε*-separated. In particular, the solution sets of* (P2) *and* (P4) *coincide.*

Solutions to (P4) can be approximated using heuristics such as path relinking [21], tabu search [21], and parallel tempering [22].

3.3 Chunk Collapsing

Having identified a maximal ε-separated subset $S \subseteq P$, we *collapse* the vertices $P \setminus S$ into S, meaning we update 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* $x \in S$ by generating a so-called Voronoi decomposition (depicted in Fig. 1a) of each chunk P, which is a partition, where each point $x \in P$ is assigned to the closest point within a subset S. More precisely, we define the sets C_x as in (1), for each $x \in S$. By construction, C_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. We then assign the coarsened chunk S a new weight function w^* defined by $w^*(x) = \sum_{y \in C_x} w(y)$. 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.

3.4 Iterative Implementation

BiDViT 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 call these iterations BiDViT *levels*. The increase of ε between BiDViT levels is realized by multiplying ε by a constant factor, denoted by α and specified by the user.

It is worth noting that, at each level, instead of proceeding with the identified representative data points, one can use the cluster centroids, allowing more-accurate data coarsening and label assignment. By generating a tree (in this case not necessarily binary), the algorithm keeps track of all BiDViT levels and which data points collapse into which centroids 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 Fig. 2 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. Analogously to agglomerative (bottom-up) hierarchical clustering methods, our algorithm begins by assigning each point its own cluster and coarsens until every single point has been assigned to one common cluster.

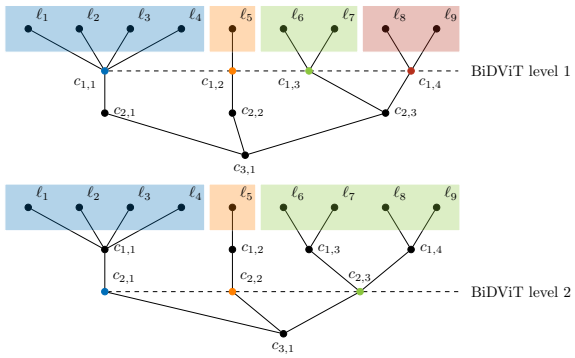


Fig. 2: A dendrogram representing the output tree of BiDViT and the encoding of the clustering assignment. The original dataset is represented by the leaves, which collapse into a single centroid after three BiDViT iterations. The first iteration ("BiDViT level 1") results in four centroids, each of which corresponds to a cluster consisting of the nodes that collapsed into it. At the next iteration, 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.

3.5 Complexity Analysis

The discussion on the order of complexity comprises two parts: data partitioning and data coarsening. Our analysis shows that the heuristic version of BiDViT has a computational complexity of $\mathcal{O}(dn\log(n/\kappa) + dn\kappa)$. Note that $\kappa \ll n$.

The order of complexity of the partitioning procedure is $\mathcal{O}(dn\log(n/\kappa))$. To see this, 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 has a complexity of $\mathcal{O}(dn_i)$ and determining the median of this dimension can be achieved in $\mathcal{O}(n_i)$ via the "median of medians" algorithm. Having computed the median, one can construct two chunks of equal size in linear time. Since $\sum_{1 \leq i \leq 2^{s-1}} n_i = n$, a partitioning step is $\mathcal{O}(dn\log(n/\kappa))$. Any division of a chunk is independent of the other chunks at a given stage; thus, this procedure can benefit from distributed computing.

The order of complexity for the collapsing process is $\mathcal{O}(dn\kappa)$, as computing the neighbourhood matrix of a chunk is $\mathcal{O}(d\kappa^2)$ and the heuristic selection procedure is $\mathcal{O}(\kappa^2)$. The number of chunks is bounded from above by $\lceil n/\kappa \rceil$. This yields a complexity of $\mathcal{O}((n/\kappa)(d\kappa^2 + \kappa^2)) = \mathcal{O}(dn\kappa)$. As data coarsening in each chunk is independent, with $\lceil n/\kappa \rceil$ parallel processors available the complexity reduces to $\mathcal{O}(d\kappa^2)$.

3.6 Relation to the MWIS Problem

The process of identifying a maximal ε-separated set of maximum weight 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. A set of vertices $S \subseteq V$ 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 ε apart. The MWIS problem can be expressed as

$$\underset{S\subseteq V}{\text{maximize}} \sum_{v\in S} w(v) \quad \text{subject to} \quad \sum_{v\in S} \sum_{u\in S:u\sim v} 1 = 0, \tag{P5}$$

and is NP-complete for a general weighted graph [12], yet for specific graphs, there exist polynomial-time algorithms [23,24]. Note that the QUBO formulation of the MWIS problem in [20, 25] is related to the QUBO formulation for the coarsening step in BiDViT.

If all weights are positive, a *maximum* weighted independent set is necessarily a *maximal* independent set. 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 ε-separated subset is ε-dense, justifying that BiDViT identifies point clouds representative of the original dataset.

4 Results

To demonstrate the efficiency and robustness of the proposed approach, we present the results of applying BiDViT to multiple datasets. The datasets used are the MNIST dataset of handwritten digits [26], a two-dimensional version of MNIST which we obtained by using t-SNE [27], two synthetic grid datasets in two and in three dimensions, and a dataset called Covertype [28], which contains data on forests in northern Colorado. Dataset statistics are provided in Table 1.

4.1 Low-Range Clustering Domain

Although BiDViT has been specifically designed for the extreme clustering domain, it yields desirable results for low values of k. Fig. 3 shows the clustering assignment of BiDViT on the 2D grid dataset and on MNIST. The results are obtained by manually selecting a BiDViT level. 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. However, as our algorithm is based on metric balls, and some datasets might not conform to such categorization, there are datasets for which it cannot accurately assign clusters. This is true for most clustering algorithms, as they are able to recognize only specific shapes.

4.2 Extreme Clustering Capability

To evaluate the performance of BiDViT on high-dimensional datasets in the extreme clustering range, we use the *Calinski–Harabasz score* [29] and the *Davies–Bouldin score* [30]. These clustering metrics are *internal evaluation schemes*,

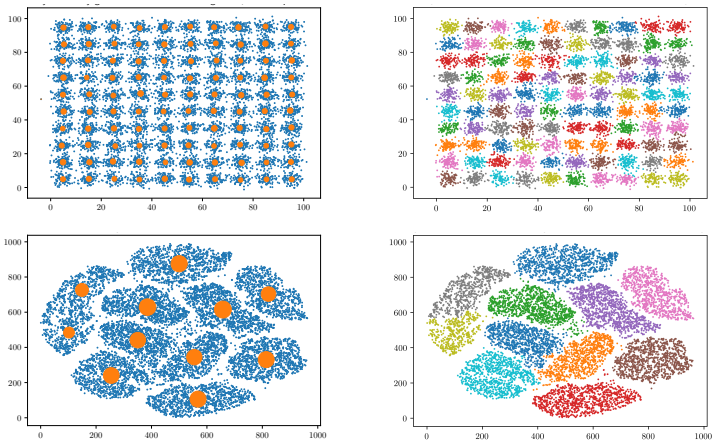


Fig. 3: Performance of BiDViT in the non-extreme clustering domain. The left-hand side figures show the original datasets (blue) with cluster centroids (orange) determined by BiDViT. On the right, colours correspond to assigned labels.

that is, their values depend solely on the clustered data without requiring the ground truth label assignment for the dataset. Such schemes must be viewed as heuristic methods: 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 internal clustering measures, see [31, 32].

Let $C_1, \dots C_{n_c}$ denote a total of n_c detected clusters within a dataset X with n data points. The Calinski–Harabasz score $\mathcal{S}_{\mathrm{CH}}$ of a clustering is defined as a weighted ratio of the *inter-cluster squared deviations* to the sum of the *intra-cluster squared deviations*. More precisely, $\mathcal{S}_{\mathrm{CH}}$ is given by

$$\mathcal{S}_{\mathrm{CH}}(C_1, \dots, C_{n_c}) = \left(\frac{n-1}{n_c-1}\right) \frac{\sum_{k=1}^{n_c} |C_k| \|c_k - c\|_2^2}{\sum_{k=1}^{n_c} \sum_{x \in C_k} \|x - c_k\|_2^2}, \tag{4}$$

where c_k, for $k = 1, \dots, n_c$ are the cluster centroids, and c is their mean. High values of $\mathcal{S}_{\mathrm{CH}}$ are indicative of a high clustering quality. The Davies–Bouldin score $\mathcal{S}_{\mathrm{DB}}$ is the average maximum value of the ratios of the pairwise sums of the intra-cluster deviation to the inter-cluster deviation. The score is defined as

$$\mathcal{S}_{\mathrm{DB}}(C_1, \dots, C_{n_c}) = \frac{1}{n_c} \sum_{k=1}^{n_c} \max_{j \neq k} \frac{S_k + S_j}{\|c_k - c_j\|_2}, \tag{5}$$

where $S_i = \sum_{x \in C_i} \|x - c_i\| / |C_i|$. Low values of $\mathcal{S}_{\mathrm{DB}}$ indicate accurate clustering.

Fig. 4 shows $\mathcal{S}_{\mathrm{CH}}$ and $\mathcal{S}_{\mathrm{DB}}$ of clustering assignments obtained with BiDViT and Mini Batch k-means clustering [33] for different values of k on the Covertype dataset. Due to their high computational complexity with respect to k, many common clustering algorithms could not be applied. Remarkably, $\mathcal{S}_{\mathrm{CH}}$ values are quite similar, indicating that the cluster assignments generated by BiDViT are

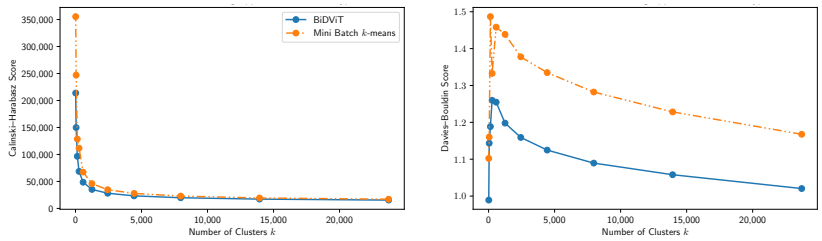


Fig. 4: Calinski–Harabasz score $\mathcal{S}_{\mathrm{CH}}$ (left) and Davies–Bouldin score $\mathcal{S}_{\mathrm{DB}}$ (right) of clustering assignments on the Covertype dataset generated by the heuristic BiDViT algorithm ($\kappa = 10^3$, $\alpha = 1, 2$) and Mini Batch k-means clustering (`batch_size` $= 50$, `max_iter` $= 10^3$, `tol` $= 10^{-3}$, `n_init` $= 1$). Whereas a higher value of $\mathcal{S}_{\mathrm{CH}}$ indicates better clustering, the opposite is the case for $\mathcal{S}_{\mathrm{DB}}$.

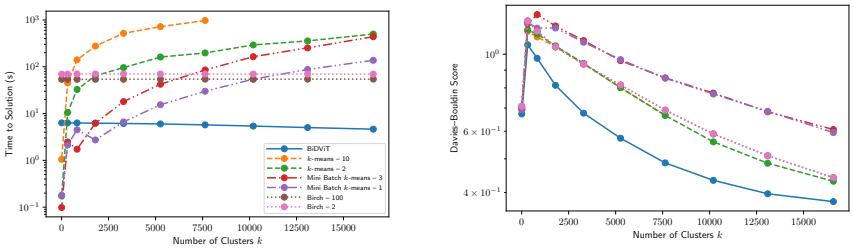


Fig. 5: Time to solution (left) and Davies–Bouldin score (right) of common clustering algorithms and the heuristic version of BiDViT on a subset of the Covertype dataset for different numbers of clusters. For k-means and Mini Batch k-means clustering, we modified the number of initializations, and for Birch clustering, it was the branching factor. These parameters resulted in a speed-up with minimum loss of quality; their values are indicated in the legend.

of comparable quality even though the runtime of our algorithm is significantly shorter. For $\mathcal{S}_{\mathrm{DB}}$, our algorithm outperforms the others for lower values of k, and is comparable for large values. One explanation for the slightly weaker performance of BiDViT with respect to $\mathcal{S}_{\mathrm{CH}}$ is that BiDViT aims to minimize the *non-squared* distances, whereas $\mathcal{S}_{\mathrm{CH}}$ rewards clustering methods that minimize *squared* distances. Similarly, this explains BiDViT's advantage for $\mathcal{S}_{\mathrm{DB}}$.

4.3 Runtime Comparison

In our experiments, we observe that, with respect to the total runtime, even the heuristic version of BiDViT restricted to a single core, outperforms common clustering methods in the extreme clustering domain. Fig. 5 shows the runtime required by different clustering algorithms for 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

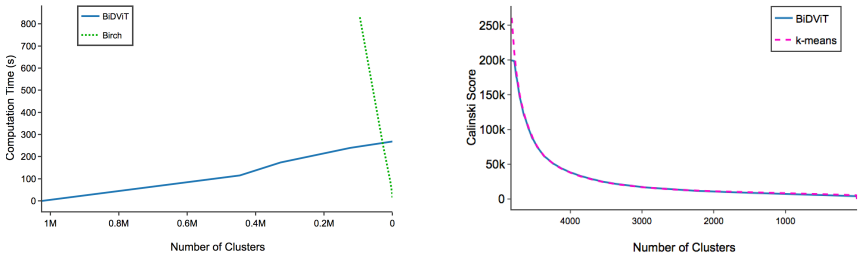


Fig. 6: Runtime and quality of results for the quantum version of BiDViT obtained using a D-Wave 2000Q quantum annealer. The figure on the left shows the computational time for the 2D gird dataset. The other figure compares the Calinski–Harabasz score of the quantum version of BiDViT and of k-means clustering on a subset of the MNIST dataset for different numbers of clusters.

entire BiDViT procedure once and then applied 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 against which it was compared, and multiple orders of magnitude faster than that of k-means and Mini Batch k-means clustering. The dataset cardinality was restricted to 20,000 points to obtain results for other methods, whereas BiDViT is capable of handling the entire dataset comprising 581,000 points.

Finally, we compared the runtime of BiDViT to *PERCH* ("Purity Enhancing Rotations for Cluster Hierarchies"), a hierarchical algorithm for extreme clustering [7]. To our knowledge, it is the only other algorithm designed to solve extreme clustering problems. We restricted both algorithms to using only a single core. Table 1 shows that BiDViT performs an order of magnitude faster than PERCH. However, they solve somewhat different problems; whereas BiDViT aims to gradually coarsen the dataset by finding ε-separated, ε-dense subsets, PERCH maximizes the *dendrogram purity*, which is a measure of the consistency of the clustering tree (see [7] for a technical definition). The clustering tree generated by PERCH is binary and thus enormous in size, allowing for much finer incremental cluster distinctions. In contrast, the tree generated by BiDViT is more compact, as 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 the opposite. We did not test these hypotheses, 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.

4.4 Results for the Quantum Version of BiDViT

We tested an early prototype of BiDViT on a D-Wave 2000Q quantum annealer, a machine that has 2048 qubits and 5600 couplers. According to D-Wave Sys-

Table 1: Dataset statistics (first table) and runtime comparison of extreme clustering algorithms in seconds (second table). "PERCH-C" (a collapsed-mode version) was run, as it performs better than standard PERCH. The parameter L sets the maximum number of leaves (see [7] for an explanation). BiDViT selected the values $\varepsilon_0 = 30$ and $\varepsilon_0 = 0.5$, such that a percentage of the nodes collapsed in the initial iteration, for the Covertype and the MNIST dataset, respectively. The mean and standard deviation were computed over five runs.

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

Algorithm specified parameters	Runtime on Dataset (seconds) Covertype	grid-3D
PERCH-C $L = \text{Inf}$	1616.45 ± 20.37	1588.10 ± 41.46
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) $\kappa = 500, \alpha = 1.2$	56.26 ± 0.62	75.22 ± 0.95

tems, 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 [13].

We observed higher-quality solutions and a significant speed-up for BiDViT, when compared to common clustering methods. Both observations are based on results shown in Fig. 6. 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 equivalent degree of speed-up.

5 Conclusion

We have developed an efficient algorithm capable of performing extreme clustering. Our complexity analysis and numerical experiments show that if the dataset cardinality and the desired number of clusters 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 ε-dense, ε-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 Thm. 2, we have identified a domain for the radius of interest such that, under a separability assumption, every solution to (P1) yields an optimal clustering assignment, justifying further investigation of this approach.

Acknowledgements

We thank Austin Wallace, Saeid Allahdadian, and Daniel Crawford for contributing to an earlier version of the algorithm. Furthermore, we want to thank Maliheh Aramon, Pooja Pandey, and Brad Woods for helpful discussions on optimization theory. Partial funding for this work was provided by the Mitacs Accelarate internship initiative.

References

1. Zhang, T., Ramakrishnan, R., Livny, M.: BIRCH: An Efficient Data Clustering Method for Very Large Databases. In: ACM Sigmod Record. vol. 25, pp. 103–114. ACM (1996)
2. Nayak, R., Mills, R., De-Vries, C., Geva, S.: Clustering and Labeling a Web Scale Document Collection using Wikipedia clusters. In: Proceedings of the 5th International Workshop on Web-scale Knowledge Representation Retrieval & Reasoning. pp. 23–30. ACM (2014)
3. de Vries, C.M., de Vine, L., Geva, S., Nayak, R.: Parallel Streaming Signature EM-tree: A Clustering Algorithm for Web Scale Applications. In: Proceedings of the 24th International Conference on World Wide Web. pp. 216–226. International World Wide Web Conferences Steering Committee (2015)
4. Wang, X.J., Zhang, L., Liu, C.: Duplicate Discovery on 2 Billion Internet Images. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops. pp. 429–436 (2013)
5. Liu, T., Rosenberg, C., Rowley, H.A.: Clustering Billions of Images with Large Scale Nearest Neighbor Search. In: Proceedings of the 8th IEEE Workshop on Applications of Computer Vision. pp. 28–. WACV '07, IEEE Computer Society, Washington, DC, USA (2007)
6. Woodley, A., Tang, L.X., Geva, S., Nayak, R., Chappell, T.: Parallel K-Tree: A multicore, multinode solution to extreme clustering. Future Generation Computer Systems (2018)
7. Kobren, A., Monath, N., Krishnamurthy, A., McCallum, A.: A Hierarchical Algorithm for Extreme Clustering. In: Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. pp. 255–264. ACM (2017)
8. Kumar, V., Bass, G., Tomlin, C., Dulny, J.: Quantum annealing for combinatorial clustering. Quantum Information Processing **17**(2), 39 (2018)
9. Merendino, S., Celebi, M.E.: A Simulated Annealing Clustering Algorithm Based On Center Perturbation Using Gaussian Mutation. In: The 26th International FLAIRS Conference (2013)
10. Kurihara, K., Tanaka, S., Miyashita, S.: Quantum Annealing for Clustering. arXiv:1408.2035 (2014)
11. Lucas, A.: Ising formulations of many NP problems. Front. Phys. **2**, 5 (2014)
12. Karp, R.M.: Reducibility among Combinatorial Problems. In: Complexity of Computer Computations, pp. 85–103. Springer (1972)
13. D-Wave Systems Inc. : The D-Wave 2000Q Quantum Computer: Technology Overview (2017), https://www.dwavesys.com/sites/default/files/D-Wave%202000Q%20Tech%20Collateral_0117F.pdf, last accessed 13 Feb. 2019

14. Fujitsu Ltd.: Digital Annealer Introduction: Fujitsu Quantum-inspired Computing Digital Annealer (2018), http://www.fujitsu.com/global/documents/digitalannealer/services/da-introduction.pdf, last accessed 13 Feb. 2019
15. Kanai, M.: Rough isometries, and combinatorial approximations of geometries of noncompact Riemannian manifolds. J. Math. Soc. Jpn. **37**(3), 391–413 (1985)
16. Coulhon, T., Saloff-Coste, L.: Variétés riemanniennes isométriques à l'infini. Rev. Mat. Iberoamericana **11**(3), 687–726 (1995)
17. Balaji, S., Swaminathan, V., Kannan, K.: Approximating Maximum Weighted Independent Set Using Vertex Support. International Journal of Computational and Mathemtical Sciences **3**(8), 406–411 (2009)
18. Hifi, M.: A genetic algorithm-based heuristic for solving the weighted maximum independent set and some equivalent problems. J. Oper. Res. Soc. **48**(6), 612–622 (1997)
19. Kako, A., Ono, T., Hirata, T., Halldórsson, M.: Approximation algorithms for the weighted independent set problem in sparse graphs. Discrete Appl. Math. **157**(4), 617–626 (2009)
20. Abbott, A.A., Calude, C.S., Dinneen, M.J., Hua, R.: A Hybrid Quantum-Classical Paradigm to Mitigate Embedding Costs in Quantum Annealing. arXiv:1803.04340 (2018)
21. Lü, Z., Glover, F., Hao, J.K.: A hybrid metaheuristic approach to solving the UBQP problem. European Journal of Operational Research **207**(3), 1254–1262 (2010)
22. Zhu, Z., Fang, C., Katzgraber, H.G.: borealis – A generalized global update algorithm for Boolean optimization problems. arXiv:1605.09399 (2016)
23. Mandal, S., Pal, M.: Maximum weight independent set of circular-arc graph and its application. Journal of Applied Mathematics and Computing **22**(3), 161–174 (2006)
24. Köhler, E., Mouatadid, L.: A linear time algorithm to compute a maximum weighted independent set on cocomparability graphs. Information Processing Letters **116**(6), 391–395 (2016)
25. Hernandez, M., Zaribafiyan, A., Aramon, M., Naghibi, M.: A Novel Graph-Based Approach for Determining Molecular Similarity. arXiv:1601.06693 (2016)
26. LeCun, Y., Cortes, C., Burges, C.J.: MNIST handwritten digit database. AT&T Labs [Online]. Available: http://yann.lecun.com/exdb/mnist (2010)
27. Maaten, L.v.d., Hinton, G.: Visualizing Data using t-SNE. Journal of Machine Learning Research **9**, 2579–2605 (2008)
28. Blackard, J.A.: UCI Machine Learning Repository" (2017), http://archive.ics.uci.edu/ml, last accessed 13 Feb. 2019
29. Caliński, T., Harabasz, J.: A Dendrite Method for Cluster Analysis. Commun. Stat. Theory Methods **3**(1), 1–27 (1974)
30. Davies, D.L., Bouldin, D.W.: A Cluster Separation Measure. IEEE Transactions on Pattern Analysis and Machine Intelligence (2), 224–227 (1979)
31. Liu, Y., Li, Z., Xiong, H., Gao, X., Wu, J.: Understanding of Internal Clustering Validation Measures. In: 2010 IEEE International Conference on Data Mining. pp. 911–916 (2010)
32. Jain, R., Koronios, A.: Innovation in the cluster validating techniques. Fuzzy Optimization and Decision Making **7**(3), 233 (2008)
33. Sculley, D.: Web-scale k-means clustering. In: Proceedings of the 19th International Conference on World Wide Web. pp. 1177–1178. ACM (2010)

Supplementary Material

Appendix A: An Alternative Coarsening Method

In certain situations, a user might not want the approximating set to be ε-separated but instead might be interested in finding an ε-dense subset with a minimum number of elements, or, more generally, the minimum cost for some *cost function* $c : X \to \mathbb{N}$. Finding such a set can be realized in a very similar way to the quantum method of BiDViT. In fact, the only modifications needed would be to Sec. 3.2 in the paper, where we introduce the concept of chunk coarsening.

Let $P = \{x^{(1)}, \ldots, x^{(n)}\}$, and let $N^{(\varepsilon)}$ and s_i, for $i = 1, \ldots, n$, be defined as in Sec. 3.2. 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 ε-dense subset $S \subseteq P$ of minimum cost can then be expressed as follows:

$$\underset{s\in\{0,1\}^n}{\text{minimize}} \sum_{i=1}^{n} s_i c_i \quad \text{subject to} \quad \sum_{j=1}^{n} N_{ij}^{(\varepsilon)} s_j \geq 1, \quad i = 1, \ldots n. \tag{P6}$$

The constraints in (P6) enforce the condition that for each solution (corresponding to a subset), every point in P is represented by at least one of the points from the selected subset. The subset will not necessarily be ε-separated, but it will be ε-dense.

In the same way that finding an ε-separated subset of maximum weight corresponds to the MWIS problem, finding an ε-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 [12]. By defining $S_j = B(x^{(j)}, \varepsilon) \cap P$ 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 (P6) 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 $\sum_{j=1}^{n} N_{ij}^{(\varepsilon)} s_j - 1 = \xi_i$. In fact, given that $s \in \{0, 1\}^n$, we can see that the ξ_i must satisfy the bounds

$$0 \leq \xi_i \leq \left(\sum_{j=1}^{n} N_{ij}^{(\varepsilon)} \right) - 1, \quad \text{for } i = 1, \ldots, n. \tag{6}$$

Therefore, by dualizing the equality constraints, (P6) can be expressed as a QUBO problem:

$$\underset{\substack{s\in\{0,1\}^n\\ \mathbf{0}\leq\xi\leq(N^{(\varepsilon)}\mathbf{1})-\mathbf{1}}}{\text{minimize}} \sum_{i=1}^{n} s_i c_i + \lambda \sum_{i=1}^{n} \left[\left(\sum_{j=1}^{n} N_{ij}^{(\varepsilon)} s_j \right) - 1 - \xi_i \right]^2 . \tag{P7}$$

We will now describe how substituting a binary encoding for each of the ξ_i, for $i = 1, \ldots, n$, in (P7) yields the desired QUBO formulation. For each $i = 1, \ldots, n$, the $(N^{(\varepsilon)}\mathbf{1})_i$ possible states of ξ_i can be encoded by $\lfloor m_i \rfloor + 1$ binary variables $b_0^{(i)}, \ldots, b_{\lfloor m_i \rfloor}^{(i)}$, where $m_i = \log_2(N^{(\varepsilon)}\mathbf{1})_i$. The encoding has the form

$$\xi_i = \sum_{k=0}^{\lfloor m_i \rfloor} b_k^{(i)} \gamma_k^{(i)}, \quad \text{for } i = 1, \ldots, n, \tag{7}$$

where $\gamma_k^{(i)} \in \mathbb{N}$ are fixed coefficients that depend solely on the bounds of (6). If we were to select $\gamma_k^{(i)} = 2^k$ for $k = 0, \ldots, \lfloor m_i \rfloor$, then, if $m_i \notin \mathbb{N}$, ξ_i could assume states that do not satisfy these bounds. We can avoid this situation by manipulating the coefficient $\gamma_{\lfloor m_i \rfloor}^{(i)}$ of the final bit $b_{\lfloor m_i \rfloor}^{(i)}$ such that $\sum_{k=0}^{\lfloor m_i \rfloor - 1} 2^k + \gamma_{\lfloor m_i \rfloor}^{(i)} = (N^{(\varepsilon)}\mathbf{1})_i - 1$. This may lead to a situation where there are multiple valid encodings for the same integer, but it will always hold that

$$0 \leq \sum_{k=0}^{\lfloor m_i \rfloor} b_k^{(i)} \gamma_k^{(i)} \leq \left(\sum_{j=1}^{n} N_{ij}^{(\varepsilon)} \right) - 1, \tag{8}$$

where $\gamma_k^{(i)} = 2^k$ for $k < \lfloor m_i \rfloor$. Substituting the binary encoding into (P7) yields the following QUBO formulation:

$$\underset{\substack{b^{(i)}\in\{0,1\}^{\lfloor m_i \rfloor + 1}\\ s\in\{0,1\}^n}}{\text{minimize}} \sum_{i=1}^{n} s_i c_i + \lambda \left[\left(\sum_{j=1}^{n} N_{ij}^{(\varepsilon)} s_j \right) - 1 - \sum_{k=0}^{\lfloor m_i \rfloor} b_k^{(i)} \gamma_k^{(i)} \right]^2 . \tag{P8}$$

One can show that the solution set of this QUBO problem is equivalent to the one for (P6) for $\lambda > n\|c\|_\infty$. We have not investigated whether this bound is sharp. Note that our QUBO formulation is similar to the one described in [11], but uses a different encoding.

The number of binary variables in the QUBO formulation of this problem depends on the binary encoding of ξ. If the vertex degree in G^ε is uniformly bounded from above by a constant $\eta > 0$, then each ξ_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 $n(1 + \lfloor \log_2(\eta) \rfloor)$. In the worst case, that is, when there is vertex that is a neighbour of every other vertex, the polynomial would still comprise fewer than $n(1 + \lfloor \log_2(n) \rfloor)$ variables.

Appendix B: Proof of Theorem 3

Proof. We generalize the proof of [20, Thm. 1] and show that every solution s to (P4) satisfies the separation constraint $\sum_{i=1}^{n}\sum_{j>i} s_i N_{ij}^{(\varepsilon)} s_j = 0$. Assuming, in contradiction, that the opposite were to be the case, we could find a solution s and indices k and ℓ such that $1 \leq k < \ell \leq n$ and $s_k = s_\ell = N_{k\ell}^{(\varepsilon)} = 1$. Let e_k denote the k-th standard unit vector, and define $v = s - e_k$. Then,

$$v^T Q v = s^T Q s - e_k^T Q s - s^T Q e_k + e_k^T Q e_k \tag{9}$$

$$= s^T Q s - \sum_{j>k}^{n} s_j Q_{kj} - \sum_{i<k}^{n} s_i Q_{ik} - Q_{kk} \tag{10}$$

$$= s^T Q s - \sum_{i\neq k} s_i \lambda_{\sigma(i,k)} N_{ik}^{(\varepsilon)} + w_k, \tag{11}$$

where $\sigma : \mathbb{N}^2 \to \mathbb{N}^2$, defined by $\sigma(i,k) = (\min(i,k), \max(i,k))$, orders the index accordingly. This technicality is necessary, as we defined λ_{ij} for only $1 \leq i < j \leq n$. Since $N_{k\ell}^{(\varepsilon)} = s_\ell = 1$, we have $\sum_{i\neq k} s_i \lambda_{\sigma(i,k)} N_{ik}^{(\varepsilon)} \geq \lambda_{\sigma(\ell,k)}$, and thus

$$v^T Q v \leq s^T Q s - \lambda_{k\ell} + w_k. \tag{12}$$

Therefore, as $\lambda_{k\ell} > \max\{w_k, w_\ell\} \geq w_k$, it holds that $v^T Q v < s^T Q s$, which is absurd as, by assumption, s solves (P4).

We now show that the solution sets of (P2) and (P4) coincide. Note that (P2) is equivalent to the optimization problem

$$\underset{s\in\{0,1\}^n}{\text{minimize}}\,,\ -s^T w \quad \text{subject to} \quad s^T \overline{N}^{(\varepsilon)} s = 0. \tag{P9}$$

Let s_1 and s_2 be solutions to (P9) and (P4), respectively. We denote the objective functions by

$$p_1(s) = -s^T w \quad \text{and} \quad p_2(s) = -s^T w + s^T \left(\Lambda \circ \overline{N}^{(\varepsilon)}\right) s, \tag{13}$$

where Λ is the matrix defined by $\Lambda_{ij} = \lambda_{ij}$ for $1 \leq i < j \leq n$, and zero otherwise, and the term $\Lambda \circ \overline{N}^{(\varepsilon)} \in \mathbb{R}^{n\times n}$ denotes the Hadamard product of the matrices Λ and $\overline{N}^{(\varepsilon)}$, given by $\left(\Lambda \circ \overline{N}^{(\varepsilon)}\right)_{ij} = (\Lambda)_{ij}\left(\overline{N}^{(\varepsilon)}\right)_{ij} = \lambda_{ij}\overline{N}_{ij}^{(\varepsilon)}$. Then, as $\lambda_{ij} > \max\{w_i, w_j\}$ for $1 \leq i < j \leq n$, by the observation above, both s_1 and s_2 satisfy the separation constraint. Since s and $\overline{N}^{\varepsilon}$ are coordinate-wise non-negative and $\lambda_{ij} > \min_{k=1,\ldots,n} w_k > 0$ for $1 \leq i < j \leq n$, it holds that

$$s^T \overline{N}^{(\varepsilon)} s = 0 \quad \Leftrightarrow \quad s^T \left(\Lambda \circ \overline{N}^{(\varepsilon)}\right) s = 0, \tag{14}$$

that is, if s satisfies the assignment constraint, then $p_2(s) = -s^T w = p_1(s)$. Using this observation, and that s_1 and s_2 minimize p_1 and p_2, respectively, we have

$$p_1(s_1) \leq p_1(s_2) = p_2(s_2) \leq p_2(s_1) = p_1(s_1). \tag{15}$$

Hence, the inequalities in (15) must actually be equalities; thus, the solution sets of the optimization problems coincide. □

Appendix C: The Heuristic Algorithm

Algorithm 1: Heuristic Method

input : $P = \{x^{(1)}, \ldots, x^{(n)}\}$ is a data chunk; w is a weight vector; ε is a radius of interest

output: S^* is a maximal ε-separated subset with high (but not necessarily maximum) weight

$I \leftarrow \{1, \ldots, n\}$
$S^* \leftarrow \emptyset$
compute the ε-neighbourhood matrix $N^{(\varepsilon)}$
while $I \neq \emptyset$ **do**
    Select $\ell \in \arg\max_{j \in I}\{w_j\}$ uniformly at random
    $I \leftarrow \{j \in I \ : \ N^{(\varepsilon)}_{\ell,j} = 0\}$
    $S^* \leftarrow S^* \cup \{x^\ell\}$
end
return S^*