Graph sketching-based Space-efficient Data Clustering

Anne Morvan $^{1,2}$, Krzysztof Choromanski$^3$, Cédric Gouy-Pailler$^1$, and Jamal Atif$^2$

$^1$CEA, LIST, 91191 Gif-sur-Yvette, France
$^2$Université Paris-Dauphine, PSL Research University, CNRS, LAMSADE, 75016 Paris, France
$^3$Google Brain Robotics, New York, USA

February 24, 2017

Abstract

In this paper, we address the problem of recovering arbitrary-shaped data clusters from datasets while facing high space constraints, as this is for instance the case in the Internet of Things environment when analysis algorithms are directly deployed on resources-limited mobile devices collecting the data. We present DBMSTClu a new density-based non-parametric method working on a limited number of linear measurements i.e. a sketched version of the dissimilarity graph $G$ between the $N$ objects to cluster. Unlike $k$-means, $k$-medians or $k$-medoids algorithms, it does not fail at distinguishing clusters with particular structures. No input parameter is needed contrarily to DBSCAN or the Spectral Clustering method. DBMSTClu as a graph-based technique relies on the dissimilarity graph $G$ which costs theoretically $O(N^2)$ in memory. However, our algorithm follows the dynamic semi-streaming model by handling $G$ as a stream of edge weight updates and sketches it in one pass over the data into a compact structure requiring $O(N \text{polylog}(N))$ space. Thanks to the property of the Minimum Spanning Tree (MST) for expressing the underlying structure of a graph, our algorithm successfully detects the right number of non-convex clusters by recovering an approximate MST from the graph sketch of $G$. We provide theoretical guarantees on the quality of the clustering partition and also demonstrate its advantage over the existing state-of-the-art on several datasets.

1 Introduction

Clustering is one of the principal data mining tasks consisting in grouping related objects in an unsupervised manner. It is expected that objects belonging to the same cluster are more similar to each other than to objects belonging to different clusters. There exists a variety of algorithms performing this task. Methods like $k$-means [Lloyd 1982], $k$-medians [Jain and Dubes 1988] or $k$-medoids [Kaufman and Rousseeuw 1987] are useful unless the number and the shape of clusters are unknown which is unfortunately often the case in real-world applications. They are typically unable to find clusters with a non-convex shape. Although DBSCAN [Ester et al. 1996] does not have these disadvantages, its resulting clustering still depends on the chosen parameter values.

One of the successful approaches relies on a graph representation of the data. Given a set of $N$ data points $\{x_1, \ldots, x_N\}$, a graph can be built based on the dissimilarity of data where points of the dataset are the vertices and weighted edges express distances between these objects. Besides, the dataset can be already a graph $G$ modeling a network in many fields, such as bioinformatics - where gene-activation dependencies are described through a network - or social, computer, information, transportation network...
analysis. The clustering task consequently aims at detecting clusters as groups of nodes that are densely connected with each other and sparsely connected to vertices of other groups. In this context, Spectral Clustering (SC) [Nascimento and de Carvalho 2011] is a popular tool to recover clusters with particular structures for which classical $k$-means algorithm fails. When dealing with large scale datasets, a main bottleneck of the technique is to perform the partial eigendecomposition of the associated graph Laplacian matrix, though. An other inherent difficulty is to handle the huge number of nodes and edges of the induced dissimilarity graph: storing all edges can cost up to $O(N^2)$ where $N$ is the number of nodes. Over the last decade, it has been established that the dynamic streaming model (Muthukrishnan [2005]) associated with linear sketching techniques (Ahn et al. [2012a]) - also suitable for distributed processing -, is a good way for tackling this last issue. The addressed problem in this paper falls within the frame of storage limits allowing $O(N \text{polylog}(N))$ space complexity but not $O(N^2)$.

**Contributions** The new clustering algorithm DBMSTClu presented in this paper brings a solution to the following issues: 1) detecting arbitrary-shaped data clusters, 2) with no parameter, 3) in a space-efficient manner by working on a limited number of linear measurements, a sketched version of the dissimilarity graph $G$. To deal with the space constraints, DBMSTClu follows the dynamic semi-streaming model by handling $G$ as a stream of edge weight updates and sketches it in only one pass over the data into a compact structure taking $O(N \text{polylog}(N))$ space. Thanks to the property of the Minimum Spanning Tree (MST) for expressing the underlying structure of a graph, DBMSTClu automatically identifies the right number of non-convex clusters by recovering an approximate MST from the graph sketch of $G$.

The remaining of this paper is organized as follows. In Section 2 the related work about graph clustering, sketching and other space-efficient clustering algorithms is described. Section 3 gives fundamentals of the sketching technique that can be used to obtain an approximate MST of the dissimilarity graph. DBMSTClu (DB for Density-Based), the proposed MST-based algorithm for clustering is then explained in Section 4. Section 5 presents the experimental results comparing the proposed clustering algorithm to other existing methods. Finally, Section 6 concludes the work and discusses future directions.

## 2 Related work

**General graph clustering** The approach of graph representation of the data leads to the extensive literature over graph clustering - related to graph partitioning and community detection - (Schaeffer 2007). A widely used method to detect communities is the Edge Betweenness Clustering (Girvan and Newman 2002) which is a hierarchical divisive algorithm removing iteratively edges with the highest edge betweenness. This method can take into account the edge weights but has a high time complexity: $O(|E|^2|V|)$ where $|E|$ and $|V|$ denote respectively the cardinality of the set of edges $E$ and vertices $V$.

From the clustering methods point of view, DenGraph (Falkowski et al. 2007) proposes a graph version of DBSCAN which is able to deal with noise while work from Ailon et al. [2013] focuses on the problem of recovering clusters with considerably dissimilar sizes. Recent works include also approaches from convex optimization using low-rank decomposition of the adjacency matrix (Oymak and Hassibi 2011, Chen et al. 2012a,b 2014a,b). These methods bring theoretically guarantees about the exact recovery of the ground truth clustering for the Stochastic Block Model (Holland et al. 1983, Condon and Karp 2001, Rohe et al. 2011) but demand to compute the eigendecomposition of a $N \times N$ matrix (resp. $O(N^3)$ and $O(N^2)$ for time and space complexity). Moreover they are restricted to unweighted graphs (weights in the work from Chen et al. 2014b are about uncertainty of existence of an edge, not a distance between points).

**MST-based graph clustering** The MST is known to help recognizing clusters with arbitrary shapes. Clustering algorithms from this family identify clusters by performing suitable cuts among the MST edges. The first one Standard Euclidean MST (SEMSST) is from Zahn 1971 and given a number of expected clusters, consists in deleting the heaviest edges from the Euclidean MST of the considered graph but this completely fails when the intra-cluster distance is lower than the inter-clusters one. For decades since
MST-based clustering methods (Asano et al. [1988], Grygorash et al. [2006]) have been developed and can be classified into the group of density-based methods. MSDR (Grygorash et al. [2006]) relies on the mean and the standard deviation of edge weights within clusters but will encourage clusters with points far from each other as soon as they are equally "far". Moreover, it does not handle clusters with less than three points. In practice, MST-based clustering algorithms have been successfully applied in bioinformatics (Xu et al. [2002]) and image color segmentation (Grygorash et al. [2006]).

Graph sketching The work from Ahn et al. [2012a] is at the origin of graph sketches study by giving dynamic semi-streaming methods for recovering different properties of graphs (MST, connectivity, bipartiteness, etc.) and has lead to numerous works since, with among others Ahn et al. [2012b, 2013], Bhattacharya et al. [2015], Huang and Peng [2016], Bandyopadhyay et al. [2016].

Space-efficient clustering algorithms Streaming k-means (Ailon et al. [2009]) is a one-pass streaming method for the k-means problem but still fails to detect clusters with non-convex shape since only the centroid point of each cluster is stored. This is not the case of CURE algorithm (Guha et al. [2001]) which represents each cluster as a random sample of data points contained in it but this offline method has a prohibitive time complexity of $O(N^2\log(N))$ not suitable for large datasets. More time-efficient, CluStream (Aggarwal et al. [2003]) and DenStream (Cao et al. [2006]) create microclusters based on local densities in an online fashion and aggregate them later to build bigger clusters in offline steps. Though, only DenStream can capture non-spherical clusters but needs parameters like DBSCAN from which it is inspired.

3 Context, notations and graph sketching preprocessing step

Consider a dataset with $N$ points. Either the underlying network already exists, or it is assumed that a dissimilarity graph $G$ between points can be built where points of the dataset are the vertices and weighted edges express distances between these objects. For instance, this can be the Euclidean distance. In both cases, the graph on which the proposed DBMSTClu algorithm is applied should follow this definition:

**Definition 3.1 (graph $G = (V, E)$)** A graph $G = (V, E)$ consists in a set of vertices or nodes $V$ and a set of edges $E \subseteq V \times V$. No attributes are assigned to nodes or edges. The graph is undirected but weighted. The weight $w$ on an edge between node $i$ and $j$ - if this edge exists - corresponds to the normalized predefined distance between $i$ and $j$, s.t. $0 < w \leq 1$. $|V| = N$ and $|E|$ stand respectively for the cardinality of sets $V$ and $E$. $E = \{e_1, \ldots, e_M\}$ and for all edge $e_i$ is assigned a weight $w_i$ representing a distance between two vertices. In the sequel, $E(G)$ is used to describe the set of edges of a graph $G$.

Our streaming clustering algorithm is divided into two main steps: 1) From the stream of edge weights, a sketch of the graph is built and then an approximate MST is retrieved exclusively from it. 2) Freely of any parameter, the nodes clustering phase is finally performed from the recovered approximate MST. Note that the two steps are independent. The use of the graph sketching technique is motivated here to give a space-efficient trick to compute a MST (which is approximate) but another method can be used instead since DBMSTClu takes as input only a MST.

3.1 Streaming graph sketching and recovery of an approximate MST

Processing data in the dynamic streaming model (Muthukrishnan [2005]) for graph sketching implies the following: 1) The graph should be handled as a stream $s$ of edge weight updates: $s = (a_1, \ldots, a_j, \ldots)$ where $a_j$ is the $j$-th update in the stream corresponding to the tuple $a_j = (i, w_{old,i}, \Delta w_i)$ with $i$ denoting the index of the edge to update, $w_{old,i}$ its previous weight and $\Delta w_i$ the update to perform. Thus, after reading $a_j$ in the stream, the $i$-th edge is assigned the new weight $w_i = w_{old,i} + \Delta w_i \geq 0$. 2) The method should make only one pass over this stream. 3) Edges can be both inserted or deleted (turnstile model),
i.e weights can be increased or decreased (but have always to stay positive). So weights change regularly, as in social networks where individuals can be friends for some time then not anymore.

Algorithm from [Ahn et al., 2012a] satisfies these conditions and is used here to produce in an online fashion a limited number of linear measurements summarizing edge weights of $G$, as new data $a_j$ are read from the stream $s$ of edge weight updates. Its general principle is briefly described here. For a given small $\epsilon_1$, $G$ is seen as a set of unweighted subgraphs $G_k$ containing all the edges with weight lower than $(1 + \epsilon_1)^k$, hence $G_k \subset G_{k+1}$. The $G_k$ are embodied as $N$ virtual vectors $v^{(i)} \in \{-1, 0, 1\}^M$ for $i \in [N]$ expressing for each node the belonging to an existing edge: for $j \in [M]$, 0 if node $i$ is not in $e_j$, 1 (resp. -1) if $e_j$ exists and $i$ is its left (resp. right) node. All $v^{(i)}$ are described at $L$ different "levels", i.e. $L$ copies of the true vectors are made with some entries randomly set to zero s.t. the $v^{(i),l}$ get sparser as the corresponding level $l \in [L]$ increases. The $v^{(i),l}$ for each level are explicitly coded in memory by three counters: $\phi = \sum_{j=1}^M v_j^{(i),l}$; $\iota = \sum_{j=1}^M j \cdot v_j^{(i),l}$; $\tau = \sum_{j=1}^M v_j^{(i),l} \cdot z^j$ mod $p$, with $p$ a suitably large prime and $z \in \mathbb{Z}_p$. The resulting compact data structure further named sketch enables to draw almost uniformly at random a nonzero weighted edge among $G_k$ at any time among the levels vectors $v^{(i),l}$ which are 1-sparse (with exactly one nonzero coefficient) thanks to $\ell_0$-sampling (Cormode and Firmani, 2014):

**Definition 3.2 ($\ell_0$-sampling)** An $(\epsilon, \delta)$ $\ell_0$-sampler for a nonzero vector $x \in \mathbb{R}^n$ fails with a probability at most $\delta$ or returns some $i \in [n]$ with probability $(1 \pm \epsilon) |\supp(x)|^{-1}$ where $\supp(x) = \{i \in [n] \mid x_i \neq 0\}$.

The sketch requires $O(N \log(N))$ space (or more precisely $O(\log^3(N))$). It follows that the sketching is technically semi-streamed but in practice only one pass over the data is needed and the space cost is significantly lower than the theoretical $O(N^2)$ bound. Note that the time cost for each update of the sketch is $\log(N)$.

In the same work, [Ahn et al., 2012a] also proposed an algorithm to compute approximately in a single-pass the weight of a MST by appropriate samplings from the sketch with $O(N \log(N))$ time complexity. Here an extended method is applied for obtaining rather an approximate MST - and not simply its weight - by registering edges as they are sampled. Referring to the proof of Lemma 3.4 in [Ahn et al., 2012a], the approach is simply justified by applying Kruskal’s algorithm where edges with lower weights are first sampled.

Note that the term MST is kept in the whole paper for the sake of simplicity, but the sketching technique so our algorithm also enables to recover a Minimum Spanning Forest if the initial graph is disconnected.

4 The proposed MST-based graph clustering method: DBMSTClu

4.1 Principle

Let us consider a dataset with $N$ points. After the sketching phase, an approximate MST further named $T$ has been obtained with $N - 1$ edges s.t. $\forall i \in [N - 1], 0 < w_i \leq 1$. Our density-based clustering method DBMSTClu exclusively relies on this object by performing some cuts among the edges of the tree s.t. $K - 1$ cuts result in $K$ clusters. Note that independently of the technique used to obtain a MST (the sketching method is just a possible one), the space complexity of the algorithm is $O(N)$ which is better than the $O(N^2)$ of SC. The time complexity of DBMSTClu is $O(NK)$ which is clearly less than the $O(N^3)$ one implied by SC. After a cut, obtained clusters can be seen as subtrees of the initial $T$ and the analysis of their qualities is only based on edges contained in those subtrees. For all clusters $C_i$, $i \in [K]$, we denote the corresponding subtree of $T$, $S_i$. In the sequel, for instance the maximal edge of a cluster will refer to the edge with the maximum weight from the subtree associated to the cluster.

Our algorithm is a parameter-free divisive top-down procedure (as opposed to an agglomerative clustering algorithm): it starts from one cluster containing the whole dataset and at each iteration, a cut

---

4 In the sequel, for a given integer $a$, $[a] = \{1, \ldots, a\}$. 4
Figure 1: Illustration of the introduced definitions with $N = 12$, $K = 3$ for cluster $C_1$ represented by the subtree $S_1$ of $T$ (dashed-circled set in the middle).

corresponding to the one maximizing some criterion is performed. The criterion used for identifying the best cut to do (if any should be made) at a given stage is a measure of the validity of the resulting clustering partition. This is a function of two positive quantities defined below: Dispersion and Separation of one cluster. The quality of a given cluster is then measured from Dispersion and Separation while the quality of the clustering partition results from the weighted average of all cluster validity indices. Finally, all those measures are based on the value of edge weights and the two latter ones lie between $-1$ and $1$.

**Definition 4.1 (Cluster Dispersion)** The Dispersion of a cluster $C_i$ (DISP) represented by the subtree $S_i$ of MST $T$ is defined as the maximum edge weight of $S_i$. If the cluster is a singleton (i.e. contains only one node), the associated Dispersion is set to $0$. More formally, with $S_i$ the subtree of $T$ representing $C_i$:

$$\forall i \in [K], \text{DISP}(C_i) = \begin{cases} \max_{j, e_j \in S_i} w_{e_j} & \text{if } |E(S_i)| \neq 0 \\ 0 & \text{otherwise.} \end{cases} \quad (1)$$

**Definition 4.2 (Cluster Separation)** The Separation of a cluster $C_i$ (SEP) is defined as the minimum distance between the nodes of $C_i$ and the ones of all other clusters $C_j, j \neq i, 1 \leq i, j \leq K, K \neq 1$ where $K$ is the total number of clusters. In practice, it corresponds to the minimum weight among all already cut edges from $T$ comprising a node from $C_i$. If $K = 1$, the Separation is set to $1$. More formally, with $\text{Cuts}(C_i)$ denoting edges incident to $C_i$,

$$\forall i \in [K], \text{SEP}(C_i) = \begin{cases} \min_{j, e_j \in \text{Cuts}(C_i)} w_{e_j} & \text{if } K \neq 1 \\ 1 & \text{otherwise.} \end{cases} \quad (2)$$

Fig. 1 sums up introduced definitions. The higher the Separation, the farther is the cluster separated from the other clusters, while low values suggest that the cluster is close to the nearest one.

**Definition 4.3 (Validity Index of a Cluster)** The Validity Index of a cluster $C_i$, $1 \leq i \leq K$ is defined as:

$$V_C(C_i) = \frac{\text{SEP}(C_i) - \text{DISP}(C_i)}{\max(\text{SEP}(C_i), \text{DISP}(C_i))} \quad (3)$$

The Validity Index of a Cluster (illustration in Fig. 2) is defined s.t. $-1 \leq V_C(C_i) \leq 1$ where $1$ stands for an optimal validity index and $-1$ for the worst. A division by zero (i.e. $\max(\text{DISP}(C_i), \text{SEP}(C_i)) = 0$) can never happen because at least Separation is positive. When Dispersion is higher than Separation, $-1 < V_C(C_i) < 0$. On the contrary, when Separation is higher than Dispersion, $0 < V_C(C_i) < 1$. So our clustering algorithm will naturally encourage clusters with a higher Separation over those with a higher Dispersion.
Figure 2: Example of the Validity Index of a Cluster with $N = 3$. For a small $\epsilon$, cutting edge with weight $\epsilon$ or 1 gives resp. left and right partitions. (left) $V_C(C_{\text{left}}^\epsilon) = \epsilon$; $V_C(C_{\text{right}}^\epsilon) = \epsilon - 1 < 0$. (right) $V_C(C_{\text{left}}^1) = 1 - \epsilon > 0$; $V_C(C_{\text{right}}^1) = 1$. Thus, the right partition, for which validity indices of each cluster are positive, is preferred.

Definition 4.4 (Validity Index of a Clustering Partition) The Density-Based Validity Index of a Clustering partition $\Pi = \{C_i\}_{1 \leq i \leq K}$, $\text{DBCVI}(\Pi)$ is defined as the weighted average of the Validity Indices of all clusters in the partition where $N$ is the number of points in the dataset.

$$\text{DBCVI}(\Pi) = \frac{1}{N} \sum_{i=1}^{K} |C_i| V_C(C_i)$$

The Validity Index of Clustering lies also between $-1$ and 1 where 1 stands for an optimal density-based clustering partition while $-1$ stands for the worst one.

The previously defined quantities are significantly distinct from the separation and sparseness defined in [Moulavi et al., 2014]. Indeed, firstly, their quantities are not well defined for special cases when clusters have less than four nodes or a partition contains only one cluster. Secondly, differentiating internal and external nodes or edges like they do does not enable to properly recover easy clusters like the three convex blobs from Fig. 5. Moreover, our DBCVI differs from the Silhouette Coefficient from [Rousseeuw, 1987]. Although this is based on close concepts like tightness and also separation, the global coefficient is based on the average values of Silhouette coefficients of each point, while our computation of DBCVI begins at the cluster level.

DBMSTClu is summarized in Algorithm 1. It starts from a partition with one cluster containing the whole dataset whereas the associated initial DBCVI is set to the worst possible value: $-1$. As long as there exists a cut which makes the DBCVI greater from (or equal to) the one of the current partition, a cut is greedily chosen by maximizing the obtained DBCVI among all the possible cuts. When no direct improvement is possible, the algorithm stops. It is guaranteed that the cut edge maximizes the DBCVI at each iteration since by construction, the greedy algorithm will try each possible cut. In practice, the algorithm stops after a reasonable number of cuts, getting trapped in a local maximum corresponding to a meaningful cluster partition. This prevents from obtaining a partition where all points are in singleton clusters. Indeed, such a result ($K = N$) is not desirable, although it is optimal in the sense of the DBCVI, since in this case, $\forall i \in [K]$, $\text{DISP}(C_i) = 0$ and $V_C(C_i) = 1$. Moreover, the non-parametric characteristic helps achieving stable partitions. In Algorithm 1, performCut applies the cut of the edge in parameter w.r.t. already formed clusters. getDBCVI computes the Validation Index of the Clustering partition in parameter.

4.2 Quality of clusters

An analysis of the algorithm and the quality of the recovered clusters is given in this section. The main results are the following: 1) DBMSTClu differs significantly from the naive approach of SEMST by preferring cuts which do not necessarily correspond to the heaviest edge (Prop. 1 and 2). 2) As long as the current partition contains at least one cluster with a negative validity index, DBMSTClu will find a
Algorithm 1 Clustering algorithm DBMSTClu

1: **Input:** $T$, the MST
2: splitDBCVI ← −1.0; cut_candidate_list ← [edges($T$)]; clusters = []
3: while splitDBCVI < 1.0 do
4:   temp_cut ← None; temp_DBCVI ← splitDBCVI
5:   for each cut in cut_candidate_list do
6:     newClusters ← performCut(clusters, cut);
7:     new_DBCVI ← getDBCVI(newClusters, $T$)
8:     if new_DBCVI ≥ temp_DBCVI then
9:       temp_cut ← cut; temp_DBCVI ← new_DBCVI
10:   if temp_cut ≠ None then
11:     clusters ← performCut(clusters, temp_cut); splitDBCVI ← temp_DBCVI
12:   else
13:     break
14: return clusters, splitDBCVI

Cut improving the global index (Prop. 3). 3) Conditions are given to determine in advance if and which cut will be performed in a cluster with a positive validity index (Prop. 4 and 5). All are completely independent of the sketching phase. Prop. 1 and 2 rely on the two basic lemmas regarding the first cut in the MST:

Lemma 1 (Case of the edge with the highest weight) Let $T$ be a MST of the dissimilarity data graph. If the first cut performed by DBMSTClu from $E(T)$ is the heaviest edge, then the resulting DBCVI is positive.

**Proof 1** For the first cut, both separations of obtained clusters $C_1$ and $C_2$ are equal to the weight of the considered edge for cut. Here, this is the one with the highest weight. Thus, for $i = 1, 2$, $\text{DISP}(C_i) = \text{SEP}(C_i) \leq V_C(C_i) \geq 0$. Finally, the DBCVI of the partition, as a convex sum of two nonnegative quantities, is clearly positive.

Lemma 2 (Case of the edge with the lowest weight) Let $T$ be a MST of the dissimilarity data graph. If the first cut done by DBMSTClu from $E(T)$ is the one with the lowest weight, then the resulting DBCVI is negative.

**Proof 2** Same reasoning in the opposite case s.t. $\text{SEP}(C_i) - \text{DISP}(C_i) \leq 0$ for $i \in \{1, 2\}$.

Proposition 1 (When the first cut is not the heaviest) Let $T$ be a MST of the dissimilarity data graph with $N$ nodes. Let us consider the following specific case: all edges have a weight equal to $w$ except two edges $e_1$ and $e_2$ resp. with weight $w_1$ and $w_2$ s.t. $w_1 > w_2 > w > 0$. DBMSTClu does not cut any edge with weight $w$ and cuts $e_2$ instead of $e_1$ as a first cut iff:

$$w_2 > \frac{2n_2w_1 - n_1 + \sqrt{n_2^2 + 4w_1(n_2^2w_1 + N^2 - Nn_1 - n_2^2)}}{2(N - n_1 + n_2)}$$

(5)

where $n_1$ (resp. $n_2$) is the number of nodes in the first cluster resulting from the cut of $e_1$ (resp. $e_2$). Otherwise, $e_1$ gets cut.

**Proof 3** Let $\text{DBCVI}_1$ (resp. $\text{DBCVI}_2$) be the DBCVI after cut of $e_1$ (resp. $e_2$). As $w$ (resp. $w_1$) is the minimum (resp. maximal) weight, the algorithm does not cut $e$ since the resulting DBCVI would be negative (cf. Lemma 3) while $\text{DBCVI}_1$ is guaranteed to be positive (cf. Lemma 7). So, the choice will be between $e_1$ and $e_2$ but $e_2$ gets cut iff $\text{DBCVI}_2 > \text{DBCVI}_1$. $\text{DBCVI}_1$ and $\text{DBCVI}_2$ expressions are
simplified w.l.o.g. by scaling the weights by \( w \) s.t. \( w \leftarrow 1, w_1 \leftarrow w_1/w, w_2 \leftarrow w_2/w, \) hence \( w_1 > w_2 > 1. \) Then,

\[
DBCVI_2 > DBCVI_1 > 0 \iff \frac{n_2}{N} \left( \frac{w_2}{w_1} - 1 \right) + \left( 1 - \frac{n_2}{N} \right) \left( 1 - \frac{1}{w_2} \right) \frac{w_2}{2w_1} + \frac{n_2}{N} \left( \frac{w_2}{w_1} + \frac{1}{w_2} - 2 \right) = 0 \]

Clearly, \( \Delta = b^2 - 4ac \) is positive and \( c/a \) is negative. But \( w_2 > w_0 > 0, \) then \( w_2 > \frac{-b + \sqrt{b^2 - 4ac}}{2a} \) which gives the final result after some simplifications.

This proposition emphasizes that the algorithm is cleverer than simply cutting the heaviest edge first. Indeed, although \( w_2 < w_1, \) cutting \( e_2 \) could be preferred over \( e_1. \) Moreover, an edge with weight \( w \) s.t. \( w < w_2 < w_1 \) can never get cut at the first iteration. This proposition can seem very particular but in practice it really happens as an approximate MST with discrete rounded weights is used.

**Proposition 2 (First cut on the heaviest edge in the middle)** Let \( T \) be a MST of the dissimilarity data graph with \( N \) nodes. Let us consider the following specific case: all edges have a weight equal to \( w \) except two edges \( e_1 \) and \( e_2 \) resp. with weight \( w_1 \) and \( w_2 \) s.t. \( w_1 > w_2 > w > 0. \) Denote \( n_1 \) (resp. \( n_2 \)) the number of nodes in the first cluster resulting from the cut of \( e_1 \) (resp. \( e_2). \) In the particular case where edge \( e_1 \) with maximal weight \( w_1 \) stands between two subtrees with the same number of points, i.e. \( n_1 = N/2, \) \( e_1 \) is always preferred over \( e_2 \) as the first optimal cut.

**Proof 4** A reductio ad absurdum is made by showing that cutting edge \( e_2 \) i.e. \( DBCVI_2 > DBCVI_1 \) leads to the contradiction \( w_1/w < 1. \) With the scaling process from Prop. 1 proof:

\[
DBCVI_1 = \frac{1}{2} (1 - \frac{1}{w_1}) + \frac{1}{2} (1 - \frac{w_2}{w_1}) = 1 - \frac{1}{2w_1} - \frac{w_2}{2w_1}_1
\]

\[
DBCVI_2 = \frac{n_2}{N} \left( \frac{w_2}{w_1} - 1 \right) + \left( 1 - \frac{n_2}{N} \right) \left( 1 - \frac{1}{w_2} \right) = 1 - \frac{1}{w_2} + \frac{n_2}{N} \left( \frac{w_2}{w_1} + \frac{1}{w_2} - 2 \right)
\]

There is \( w_2 > w = 1, \) so \( \frac{1}{w_2} < 1. \) Besides \( w_2 < w_1 \) so \( \frac{w_2}{w_1} < 1 \) thus, \( A < 0. \) Let now consider w.l.o.g. that edge \( e_2 \) is on the “right side” (right cluster/subtree) of \( e_1 \) (similar proof if \( e_2 \) is on the left side of \( e_1). \) Hence, it is clear that for maximizing \( DBCVI_2 \) as a function of \( n_2, \) we need \( n_2 = n_1 + 1. \) Then,

\[
DBCVI_2 > DBCVI_1 \iff - \frac{1}{w_2} + \left( \frac{1}{2} + \frac{1}{N} \right) \left( \frac{w_2}{w_1} - 1 \right) + \left( 1 - \frac{1}{w_2} \right) > - \frac{1}{w_1} - \frac{w_2}{w_1}
\]

\[
\iff (2w_1 + \frac{1}{Nw_1} + \frac{1}{2w_1})w_2 - 1 - \frac{2}{N} + \frac{1}{2w_1} + (1 - \frac{1}{2} + \frac{1}{N}) \frac{1}{w_2} > 0
\]

\[
\iff (1 + \frac{N}{1}) w^2 < w_2 \left( \frac{1}{2} - w_1 \left( \frac{1}{2} + \frac{2}{N} \right) \right) + w_1 \left( \frac{1}{N} - \frac{1}{2} \right) > 0
\]

As \( c/a < 0 \) and \( w_2 > 0, \) \( w_2 > \frac{N}{2(N+1)} \left[ w_1 (1 + \frac{N}{2}) - \frac{1}{2} + \sqrt{\Delta} \right] \) with \( \Delta = \left( w_1 (1 + \frac{N}{2}) - \frac{1}{2} \right)^2 + 4 \left( 1 + \frac{1}{N} \right) \left( \frac{1}{2} - \frac{1}{N} \right) w_1. \) This inequality is incompatible with \( w_1 > w_2 \) since:

\[
w_1 > w_2 \iff w_1 > \frac{N}{2(N+1)} \left[ w_1 (1 + \frac{2}{N}) - \frac{1}{2} + \sqrt{\Delta} \right] \iff w_1 + \frac{1}{2} > \sqrt{\Delta}
\]

\[
\iff \frac{4}{N} w_1^2 \left( \frac{1}{2} + \frac{1}{N} \right) + \frac{4}{N} w_1 (-1 - \frac{1}{N}) < 0 \iff w_1 < 1 : ILLICIT
\]

Indeed, after the scaling process, \( w_1 = 1 = w \) is not possible since by hypothesis, \( w_1 > w. \) Finally, it is not allowed to cut \( e_2, \) the only remaining possible edge to cut is \( e_1. \)
Remark 1 Let us consider the MST in Fig. 3 with \( N = 8, w_1 = 1, w_2 = w_3 = 1 - \epsilon \), and other weights set to \( \epsilon \). Clearly, it is not preferred to cut \( e_1 \) in the middle since for \( \epsilon = 0.1 \), \( \text{DBCVI}_2 \approx 0.27 > \text{DBCVI}_1 = \epsilon = 0.1 \). So, it is a counter-example to a possible generalization of Prop. 3 where there would be more than three possible distinct weights in \( T \).

![Figure 3: Counter-example for Remark 1](image)

These last propositions hold for every iteration in the algorithm.

Proposition 3 (Fate of a cluster with a negative validity index) Let \( K = t + 1 \) be the number of clusters in the clustering partition at iteration \( t \). If for some \( i \in [K] \) we have \( V_C(C_i) < 0 \), then DBMSTClu will cut an edge at this stage.

Proof 5 Let \( i \in [K] \) s.t. \( V_C(C_i) < 0 \) i.e. \( \text{SEP}(C_i) < \text{DISP}(C_i) \). We denote \( w^i_{\text{sep}} \) the minimal weight outing cluster \( C_i \) and \( w_{\text{max}} \) the maximal weight in subtree \( S_i \) of \( C_i \) i.e. \( \text{SEP}(C_i) \leq w^i_{\text{sep}} \) and \( \text{DISP}(C_i) \leq w_{\text{max}} \). Hence, \( w^i_{\text{sep}} < w_{\text{max}} \). By cutting the cluster \( C_i \) on the edge with weight \( w_{\text{max}} \), we define \( C^l_i \) and \( C^r_i \) resp. the left and right resulting clusters. Arbitrarily we assume that the separation value \( (w^i_{\text{sep}}) \) comes from the left part of \( C_i \), but the same reasoning would apply for the right side. Fig. 3 gives an illustration of this case. After a cut on the edge with weight \( w_{\text{max}} \), separation of the left resulting cluster \( C^l_i \) remains the same since \( w^i_{\text{sep}} < w_{\text{max}} \), i.e. \( \text{SEP}(C^l_i) = \text{SEP}(C_i) = w^i_{\text{sep}} \). Then, \( \text{DISP}(C^l_i) \leq \text{DISP}(C_i) = w_{\text{max}} \) (there is equality if there is another edge with weight equal to \( w_{\text{max}} \)). So, \( V_C(C^l_i) \geq V_C(C_i) \) since the dispersion decreases.

Regarding the right side, \( \text{SEP}(C^r_i) \geq \text{SEP}(C_i) \) (equality if there is an outing edge from \( C_i \) with weight equal to \( w^i_{\text{sep}} \)). Moreover, \( \text{DISP}(C^r_i) \leq \text{DISP}(C_i) \) (equality if there is another edge with weight \( w_{\text{max}} \)). Hence \( V_C(C^r_i) \geq V_C(C_i) \). So in both cases, the validation indices of both clusters are higher than \( V_C(C_i) \) hence the DBCVI is improved. Finally, by cutting at least the heaviest edge in a cluster with a negative index, the global DBCVI is improved or at least equalized. Therefore, no cluster with a negative index will stay uncut.

As a consequence, at the end of the clustering algorithm, each cluster will have a nonnegative cluster validation index. So, at each step of the algorithm, the following bound holds for the final DBCVI:

\[
\text{DBCVI} \geq \sum_{i=1}^{K} \frac{|C_i|}{N} \max(V_C(C_i), 0).
\]

Proposition 4 (Fate of a cluster with a positive index) Let \( T \) be a MST of the dissimilarity data graph and \( C \) a cluster s.t. \( V_C(C) > 0 \) and \( \text{SEP}(C) = s \). Among the edges of the subtree of \( C \), DBMSTClu does not cut an edge \( e \) of weight \( w < s \) if both resulting clusters have at least one edge with weight greater than \( w \).

Proof 6 Let us consider clusters \( C_1 \) and \( C_2 \) resulting from the cut of edge \( e \). Assume that in the associated subtree of \( C_1 \) (resp. \( C_2 \)), there is an edge \( e_1 \) (resp. \( e_2 \)) with a weight \( w_1 \) (resp. \( w_2 \)) higher than \( w \) s.t. without loss of generality, \( w_1 > w_2 \). Since \( V_C(C) > 0 \), \( s > w_1 > w_2 > w \). But cutting edge \( e \) implies that for \( i \in \{1, 2\} \), \( \text{DISP}(C_i) > \text{SEP}(C_i) = w \), and thus \( V_C(C_i) < 0 \). Cutting edge \( e \) would therefore mean to replace a cluster \( C \) s.t. \( V_C(C) > 0 \) by two clusters s.t. for \( i \in \{1, 2\} \), \( V_C(C_i) < 0 \) which obviously decreases the current DBCVI. Thus, \( e \) does not get cut at this step of the algorithm.
Proposition 5 (Fate of a cluster with a positive validity index II) Consider a partition with \( K \) clusters s.t. a given cluster \( C_i, i \in [K] \) with \( V_C(C_i) > 0 \) is in the setting of Fig. 4 i.e. if the heaviest edge \( e \) with weight \( w_{\text{max}} \) gets cut, it gives birth to the left (resp. right) cluster \( C^l_i \) (resp. \( C^r_i \)) with \( n_1 \) points (resp. \( n_2 \)) s.t. \( \text{DISP}(C^l_i) = d_1 \), \( \text{SEP}(C^l_i) = w_{\text{sep}}^l \), \( \text{DISP}(C^r_i) = d_2 \) and \( \text{SEP}(C^r_i) = w_{\text{sep}}^r \). We suppose w.l.o.g. that \( w_{\text{sep}}^l > w_{\text{sep}}^r \). Then, \( e \) gets cut iff:

\[
\frac{n_1 d_1 + n_2 d_2}{n_1 + n_2} \leq \frac{w_{\text{max}}}{w_{\text{sep}}^r}.
\]

**Proof 7** As \( V_C(C_i) > 0 \), there is \( \text{SEP}(C_i) = w_{\text{sep}}^r > w_{\text{max}} \). Then, the DBCVI before \( (K \) clusters) and after cut of \( w_{\text{max}} \) \((K + 1 \) clusters) are:

\[
\begin{align*}
\text{DBCVI}_K &= \sum_{j \neq i} V_C(C_j) + \frac{n_1 + n_2}{N} \left( 1 - \frac{w_{\max}}{w_{\text{sep}}^r} \right) \\
\text{DBCVI}_{K+1} &= \sum_{j \neq i} V_C(C_j) + \frac{n_1}{N} \left( 1 - \frac{d_1}{w_{\text{max}}} \right) + \frac{n_2}{N} \left( 1 - \frac{d_2}{w_{\text{max}}} \right)
\end{align*}
\]

**DBMSTClu** cuts \( w_{\text{max}} \) iff \( \text{DBCVI}_{K+1} \geq \text{DBCVI}_K \). So the result after simplification.

5 Experiments

**Synthetic datasets** First experiments were performed on three classic synthetic datasets from the Euclidean space: three blobs, noisy circles and noisy moons. Each dataset contains 1000 data points in 20 dimensions: the first two dimensions are randomly drawn from predefined 2D-clusters, as shown in the following figures, while the other 18 dimensions are random Gaussian noise. The results of DBMSTClu are compared with two other algorithms: DBSCAN (Ester et al. [1996]) which competes with DBMSTClu regarding time and space complexities (resp. \( O(N^2) \) in the worst case and \( O(N) \) when the distance matrix is not stored) and a naive approach of a MST-based algorithm. The latter called Standard Euclidean MST (SEMST) (Zahn [1971]) cuts the \( K - 1 \) heaviest edges of the standard Euclidean MST given a targeted number of clusters \( K \). For DBMSTClu, a \( k \)-NN graph is built from computing the Euclidean distance between data points (\( k = 550 \) for noisy circles and moons, \( k = 350 \) for the three blobs) in order to speed up the sketching phase by discarding useless heaviest edges. Note that \( k \) should not be considered as a parameter of the clustering algorithm since it is used only for speedup purposes. Then this \( k \)-NN graph is
passed into the sketch phase to produce an approximate version of the exact MST. Fig. 5, 6 and 7 show the results when providing to DBMSTClu an approximate MST obtained from the sketch phase. The datasets were projected onto 2-dimensional spaces for helping results visualization. They were produced with a noise level such that SEMST fails and DBSCAN does not perform well without parameters optimization. In particular, for DBSCAN all the cross points correspond to noise. With the three blobs, each method performs well: they all manage to retrieve three clusters. Note that DBSCAN detects one point as noise which is considered as a member of the red cluster for DBMSTClu. With the concentric circles, SEMST does not cut on the consistent edges, hence leads to an isolated cluster with two points in blue. DBSCAN classifies the same points as noise while recovering the two circles well. Finally, DBMSTClu finds the two main clusters and also creates five singleton clusters which can be legitimately considered as noise. With noisy moons, while DBSCAN considers four outliers, DBMSTClu detects only three singletons. As theoretically proved above, experiments emphasize the fact that our algorithm is much more subtle than simply cutting the heaviest edges as the failure of SEMST shows. Moreover our algorithm exhibits an ability to detect outliers, which could be labeled as noise in a postprocessing phase. Another decisive advantage of our algorithm is the absence of any required parameters, contrarily to DBSCAN.

Table 1 shows quantitative results of the experiments: the achieved silhouette coefficient, Adjusted Rand Index (ARI) and DBCVI for the three datasets (in order blobs, noisy circles and moons). For all the indices, the higher, the better. Silhouette coefficient (between $-1$ and 1) is used to measure a clustering partition without any external information. For DBSCAN it is computed by considering noise points as singletons. We see that this measure is not very suitable for non-convex clusters like noisy circles or moons. The ARI (between 0 and 1) measures the similarity between the experimental clustering partition and the known groundtruth. DBSCAN and DBMSTClu give similar almost optimal results. Finally, the obtained DBCVIs are consistent, since the best ones are reached for DBMSTClu.
Figure 7: Noisy moons: SEMST, DBSCAN ($\epsilon = 0.15$, $minPts = 5$), DBMSTClu with an approximate MST.

| Silhouette coeff. | Adjusted Rand Index | DBCVI |
|-------------------|---------------------|-------|
| SEMST             | 0.843               | 0.159 | -0.12 1 0 0 | 0.837 0.001 0.058 |
| DBSCAN            | 0.733               | 0.023 | 0.253 0.998 0.992 0.992 0.501 -0.262 -0.113 |
| DBMSTClu          | 0.843               | -0.262 | -0.16 1 0.99 0.994 0.837 0.179 0.15 |

Table 1: Silhouette coefficients, Adjusted Rand Index and DBCVI for the blobs, noisy circles and noisy moons datasets with SEMST, DBSCAN and DBMSTClu.

**Mushroom dataset** DBMSTClu performances are also measured on the mushroom dataset. It contains 8124 records of 22 categorical attributes corresponding to 23 species of gilled mushrooms in the Agaricus and Lepiota family. 118 binary attributes are created from the 22 categorical ones, then the complete graph (about 33 millions of edges) is built by computing the Hamming distance (i.e. the number of distinct bits) between points normalized by the dimension 118. Fig. 8 shows the results for the three methods. We see that if suitable parameters are given to SEMST and DBSCAN, they are able to find 23 clusters while DBMSTClu retrieves them without tuning any parameter. The corresponding DBCVI and silhouette coefficient are resp. 0.75 and 0.47.

Figure 8: Mushroom dataset: SEMST, DBSCAN ($\epsilon = 1.5$, $minPts = 2$), DBMSTClu with an approximate MST (projection on the first three principal components).

6 Conclusion

In this paper we introduced DBMSTClu a novel space-efficient Density-Based Clustering algorithm which works only on a Minimum Spanning Tree (MST) of the dissimilarity graph $G$: the spatial cost is $O(N)$ with $N$ the number of data points. What makes the method particularly interesting is the fact that DBMSTClu
is non-parametric, unlike most existing clustering methods: it automatically determines the right number
of non-convex clusters. Although the approach is fundamentally independent from the sketching phase,
its robustness has been assessed by using as input an approximate MST of the sketched $G$ rather than
the exact MST itself. The graph sketch is computed dynamically on the fly as new edge weight updates
are received in only one pass over the data. Hence, our algorithm adapts to the semi-streaming setting
with $O(N \text{polylog } N)$ space. Our approach shows promising results, as evidenced by the experimental
part. Further work would consist in using this algorithm in privacy issues, as the lost information when
sketching might be sufficient for ensuring data privacy. Moreover, as it is already the case for the graph
sketching, it would be interesting to adapt both the MST recovery and DBMSTClu to the fully online
setting, i.e. to be able to modify dynamically current MST and clustering partition as a new edge weight
update from the stream is seen.

Acknowledgements

We would like to thank Mario Lucic for the fruitful private conversation and for coauthoring with Krzysztof
Choromanski the MSE sketching extension during his internship at Google.

References

C. C. Aggarwal, J. Han, J. Wang, and P. S. Yu. A framework for clustering evolving data streams. In
Proceedings of the 29th International Conference on Very Large Data Bases - Volume 29, VLDB '03,
pages 81–92. VLDB Endowment, 2003. ISBN 0-12-722442-4.

K. J. Ahn, S. Guha, and A. McGregor. Analyzing graph structure via linear measurements. In Proceedings
of the Twenty-third Annual ACM-SIAM Symposium on Discrete Algorithms, SODA ’12, pages 459–467,
Philadelphia, PA, USA, 2012a. Society for Industrial and Applied Mathematics.

K. J. Ahn, S. Guha, and A. McGregor. Graph sketches: Sparsification, spanners, and subgraphs. In
Proceedings of the 31st ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems,
PODS ’12, pages 5–14, New York, NY, USA, 2012b. ACM.

K. J. Ahn, S. Guha, and A. McGregor. Spectral Sparsification in Dynamic Graph Streams, pages 1–10.
Springer Berlin Heidelberg, Berlin, Heidelberg, 2013.

N. Ailon, R. Jaiswal, and C. Monteleoni. Streaming k-means approximation. In Y. Bengio, D. Schuurmans,
J. D. Lafferty, C. K. I. Williams, and A. Culotta, editors, Advances in Neural Information Processing
Systems 22, pages 10–18. Curran Associates, Inc., 2009.

N. Ailon, Y. Chen, and H. Xu. Breaking the small cluster barrier of graph clustering. CoRR,
abs/1302.4549, 2013.

T. Asano, B. Bhattacharya, M. Keil, and F. Yao. Clustering algorithms based on minimum and maximum
spanning trees. In Proceedings of the Fourth Annual Symposium on Computational Geometry, SCG ’88,
pages 252–257, New York, NY, USA, 1988. ACM.

B. Bandyopadhyay, D. Fuhr, A. Chakrabarti, and S. Parthasarathy. Topological graph sketching for
incremental and scalable analytics. In Proceedings of the 25th ACM International on Conference on
Information and Knowledge Management, CIKM ’16, pages 1231–1240, New York, NY, USA, 2016. ACM.

S. Bhattacharya, M. Henzinger, D. Nanongkai, and C. Tsourakakis. Space- and time-efficient algorithm
for maintaining dense subgraphs on one-pass dynamic streams. In Proceedings of the Forty-seventh
Annual ACM Symposium on Theory of Computing, STOC ’15, pages 173–182, New York, NY, USA,
2015. ACM.
F. Cao, M. Ester, W. Qian, and A. Zhou. Density-based clustering over an evolving data stream with noise. In *2006 SIAM Conference on Data Mining*, pages 328–339, 2006.

Y. Chen, S. Sanghavi, and H. Xu. Clustering sparse graphs. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 2204–2212. Curran Associates, Inc., 2012a.

Y. Chen, S. Sanghavi, and H. Xu. Clustering sparse graphs. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 2204–2212. Curran Associates, Inc., 2012b.

Y. Chen, A. Jalali, S. Sanghavi, and H. Xu. Clustering partially observed graphs via convex optimization. *J. Mach. Learn. Res.*, 15(1):2213–2238, Jan. 2014a.

Y. Chen, S. H. Lim, and H. Xu. Weighted graph clustering with non-uniform uncertainties. In *Proceedings of the 31st International Conference on International Conference on Machine Learning - Volume 32*, ICML’14, pages II–1566–II–1574. JMLR.org, 2014b.

A. Condon and R. M. Karp. Algorithms for graph partitioning on the planted partition model. *Random Struct. Algorithms*, 18(2):116–140, Mar. 2001.

G. Cormode and D. Firmani. A unifying framework for $l_0$-sampling algorithms. *Distributed and Parallel Databases*, 32(3):315–335, 2014. Special issue on Data Summarization on Big Data.

M. Ester, H. Kriegel, J. Sander, and X. Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. pages 226–231. AAAI Press, 1996.

T. Falkowski, A. Barth, and M. Spiliopoulou. Dengraph: A density-based community detection algorithm. In *Proceedings of the IEEE/WIC/ACM International Conference on Web Intelligence*, WI ’07, pages 112–115, Washington, DC, USA, 2007. IEEE Computer Society.

M. Girvan and M. E. J. Newman. Community structure in social and biological networks. *Proceedings of the National Academy of Sciences of the United States of America*, 99(12):7821–7826, 2002.

O. Grygorash, Y. Zhou, and Z. Jorgensen. Minimum spanning tree based clustering algorithms. In *2006 18th IEEE International Conference on Tools with Artificial Intelligence (ICTAI’06)*, pages 73–81, Nov 2006.

S. Guha, R. Rastogi, and K. Shim. Cure: An efficient clustering algorithm for large databases. *Inf. Syst.*, 26(1):35–58, 2001.

P. W. Holland, K. B. Laskey, and S. Leinhardt. Stochastic blockmodels: First steps. *Social Networks*, 5(2):109–137, June 1983.

Z. Huang and P. Peng. Dynamic graph stream algorithms in $o(n)$ space. *CoRR*, abs/1605.00089, 2016.

A. Jain and R. Dubes. *Algorithms for Clustering Data*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1988.

L. Kaufman and P. Rousseeuw. Clustering by means of medoids. *Statistical Data Analysis Based on the $L_1$-Norm and Related Methods*, pages 405–416, 1987.

S. Lloyd. Least squares quantization in pcm. *IEEE Trans. Inf. Theor.*, 28(2):129–137, Sept. 1982.

D. Moulavi, P. A. Jaskowiak, R. J. G. B. Campello, A. Zimek, and J. Sander. Density-based clustering validation. In *Proceedings of the 2014 SIAM International Conference on Data Mining, Philadelphia, Pennsylvania, USA, April 24-26, 2014*, pages 839–847, 2014.
S. Muthukrishnan. Data streams: Algorithms and applications. *Found. Trends Theor. Comput. Sci.*, 1 (2):117–236, Aug. 2005.

M. Nascimento and A. de Carvalho. Spectral methods for graph clustering - a survey. *European Journal of Operational Research*, 211(2):221–231, 2011.

S. Oymak and B. Hassibi. Finding dense clusters via "low rank + sparse" decomposition. *CoRR*, abs/1104.5186, 2011.

K. Rohe, S. Chatterjee, and B. Yu. Spectral clustering and the high-dimensional stochastic blockmodel. *Ann. Statist.*, 39(4):1878–1915, 08 2011.

P. J. Rousseeuw. Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. *Journal of Computational and Applied Mathematics*, 20:53 – 65, 1987.

S. E. Schaeffer. Survey: Graph clustering. *Comput. Sci. Rev.*, 1(1):27–64, Aug. 2007.

Y. Xu, V. Olman, and D. Xu. Clustering gene expression data using a graph-theoretic approach: an application of minimum spanning trees. 18(4):536–545, 2002.

C. T. Zahn. Graph-theoretical methods for detecting and describing gestalt clusters. *IEEE Trans. Comput.*, 20(1):68–86, Jan. 1971.