entropy

MDPI

Article

Cluster Persistence for Weighted Graphs

Omer Bobrowski [1,2] **and Primoz Skraba** [1,3,*]

[1] School of Mathematical Sciences, Queen Mary University of London, London E1 4NS, UK
[2] Viterbi Faculty of Electrical and Computer Engineering, Technion, Haifa 3200003, Israel
[3] Department for Artificial Intelligence, Jozef Stefan Institute, 1000 Ljubljana, Slovenia
* Correspondence: primoz.skraba@ijs.si

Abstract: Persistent homology is a natural tool for probing the topological characteristics of weighted graphs, essentially focusing on their 0-dimensional homology. While this area has been thoroughly studied, we present a new approach to constructing a filtration for cluster analysis via persistent homology. The key advantages of the new filtration is that (a) it provides richer signatures for connected components by introducing non-trivial birth times, and (b) it is robust to outliers. The key idea is that nodes are ignored until they belong to sufficiently large clusters. We demonstrate the computational efficiency of our filtration, its practical effectiveness, and explore into its properties when applied to random graphs.

1. Introduction

The **clustering** of data is a fundamental task in unsupervised machine learning and exploratory data analysis. It has been the subject of countless studies over the last 50 years, with many definitions and algorithms being proposed, e.g., [1,2]. **Persistent homology** [3,4] is a powerful topological tool that provides multi-scale structural information about data and networks [5–8]. Given an increasing sequence of spaces (filtration), persistent homology tracks the formation of connected components (0-dimensional cycles), holes (1-dimensional cycles), cavities (2-dimensional cycles), and their higher-dimensional extensions. The information encoded in persistent homology is often represented by a *persistence diagram*, which is a collection of points in $\mathbb{R}^2$ representing the birth and death of homology classes and providing an intuitive numerical representation for topological information (see Figure 1).

The connection between clustering and 0-dimensional persistent homology is well established in various different scenarios. Specifically, the relationship with functoriality [9,10] has been investigated, as well as how to combine persistence with density-based methods [11,12]. An important motivating factor for connecting these methods is *stability*. Namely, given small perturbations in the input data, persistent homology can provide guarantees on the number of the output clusters. One important drawback of this topological approach is that statistical tests for persistent homology and clustering based on persistence remain lacking.

Recently, for persistent homology in dimensions 1 and above (i.e., excluding connected components), persistent homology based on distance filtration has been experimentally shown to exhibit strong universal behavior [13]. Suppose we are given as the input a point cloud generated by some unknown distribution. If we compute the distance-based persistent homology, under an appropriate transformation, the distribution of persistence values has been shown to be independent of the original point cloud distribution. This phenomenon has since been used to develop a statistical test to detect statistically significant homology classes. A key point in [13] is that in order to obtain such universal behavior, the measure of persistence must be given by the value of death / birth, which makes the measure of persistence scale-invariant.

However, in distance-based filtration, 0-dimensional persistent homology (tracking clusters) does not fit into this universality framework as the birth time of every 0-dimensional homology class is set to 0. To address this issue and enable the study of

Citation: Bobrowski, O.; Skraba, P. Cluster Persistence for Weighted Graphs. *Entropy* **2023**, *25*, 1587. https://doi.org/10.3390/e25121587

Academic Editor: Gergely Palla

Received: 30 September 2023
Revised: 9 November 2023
Accepted: 22 November 2023
Published: 26 November 2023

universality in the context of clustering, we introduce a new filtration approach, which we call k-cluster filtration. This is a novel non-local construction, where vertices only become 'alive' once they belong to a sufficiently large cluster. In other words, while traditional persistent homology considers every vertex as an individual cluster and tracks its evolution, k-cluster filtration only considers components with k or more vertices as 'meaningful' clusters.

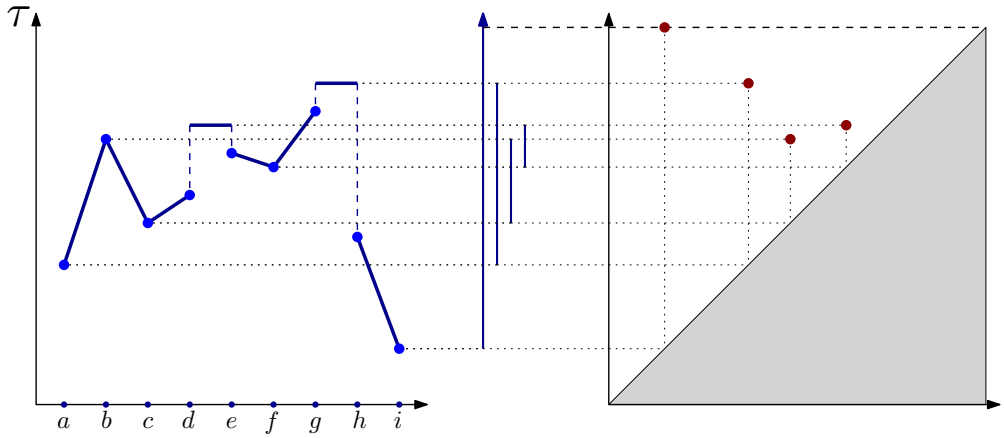

Figure 1. An example of graph filtration on a line graph. The filtration values of the vertices are given by τ (the y-axis). The filtration value of each edge is taken as the highest value between its plotted endpoints. The bars in the middle represent the tracking of the components. The vertices that are local minima, i.e., a, c, f, and i, generate new components; so, $\tau(a)$, $\tau(c)$, $\tau(f)$, and $\tau(i)$ correspond to birth times. The first merge occurs at $\tau(b) = \tau((a,b)) = \tau((b,c))$, merging $\{a\}$ with $\{c,d\}$. In this case, we declare the latter as dead since $\tau(a) < \tau(c)$. Next, at $\tau((d,e))$, the components $\{a,b,c,d\}$ and $\{e,f\}$ are merged and the latter dies. Finally, at $\tau((g,h))$, the components $\{a,b,c,d,e,f,g\}$ and $\{h,i\}$ are merged and the former dies. The component containing i has the earliest birth time and is thus declared to be infinite.

We note that while the motivation for this new filtration method is to study distance-based filtrations, k-cluster filtration can be constructed over any weighted graph. It generally provides two key advantages over traditional filtration. Firstly, it results in 'richer' persistence diagrams, in the sense that components have non-trivial birth times. This improves our ability to compare between different features within the same diagram or across different diagrams. Secondly, k-cluster filtration provides a more 'focused' view of connected components by discarding those that are considered small (determined by application). In particular, it allows us to remove outliers from persistence diagrams.

This paper is organized as follows: Section 2 provides essential background information about persistent homology; Section 3 introduces k-cluster filtration and presents some preliminary properties; Section 4 provides an algorithm for computing the filtration function and its corresponding persistence diagram in a single pass; Section 5 demonstrates some experimental results comparing our clustering method to some other approaches; and finally, Section 6 discusses some probabilistic aspects of this filtration method by comparing it to known properties of random graphs and simplicial complexes.

2. Graph Filtration and Persistent Homology

We first introduce the required topological notions. As we focus on the special case of graphs and connected components (0-dimensional homology), we restrict our definitions to this case. For the general definition of homology, we refer the reader to [14,15].

Let $G = (V, E)$ be an undirected graph. Our main object of study is *graph filtration*, i.e., an increasing sequence of graphs. This can be constructed by defining a function $\tau : (V \cup E) \to [0, \infty)$, with the restriction that if $e = (u, v) \in E$, then $\tau(e) \geq \max(\tau(u), \tau(v))$.

This restriction ensures that the sublevel sets of τ define a subgraph. The filtration $\{\mathcal{G}_t\}_{t\geq 0}$ is then defined as

$$\mathcal{G}_t = \{\sigma \in V \cup E : \tau(\sigma) \leq t\}.$$

As we increase t from 0 to ∞, we can track the connected components of $\mathcal{G}_t$ as they appear and merge, which are referred to as *births* and *deaths*, respectively. When two components merge, we use the 'elder rule' to determine that the component that was born last is the one that dies. Note that at least one component has an infinite death time in any graph filtration. We refer the reader to [3] for an in-depth introduction to persistent homology.

These birth–death events can be tracked using an algebraic object called the 0-dimensional persistent homology group. Its most common visualization is a *persistence diagram*, which is a collection of points in $\mathbb{R}^{2+}$ where each point corresponds to a single connected component. The coordinates of a point encode its information, with the x-coordinate representing the birth time and the y-coordinate representing the death time. An example of a function on a line graph is shown in Figure 1. Note that one component is infinite, which is denoted by the dashed line at the top of the diagram.

In a more general context, given the filtration of higher-dimensional objects (e.g., simplicial complexes), we can also study k-dimensional persistent homology. This object tracks the formation of k-dimensional cycles (various types of holes) and its definition is a natural extension of the 0-dimensional persistent homology we study here. However, this is beyond the scope of this paper and we refer the reader to [3] for more information.

3. k-Cluster Filtration

Let $G = (V, E, W)$ be an undirected weighted graph. When computing 0-dimensional persistent homology, the filtration values are commonly taken to be $\tau(v) = 0$ for all $v \in V$ and $\tau(e) = W(e)$ for all $e \in E$. We denote this filtration by $\mathcal{G}_t^*$. In other words, we assume all vertices are present at time zero and that edges are gradually added according to the weight function W. This has been common practice in almost all studies in the TDA literature, particularly in geometric settings where W represents the distance between points, i.e., geometric graphs, which are the skeleton of two common constructions: the Čech and Vietoris–Rips complexes [3]. While in many models, this choice of τ seems reasonable, it has two significant drawbacks:

- The produced persistence diagrams are *degenerate* as the birth time of every 0-cycle is $t = 0$, which significantly reduces the amount of information we can extract from the diagrams;
- The generated persistence diagrams are *superfluous* in the sense that they contain a point for each vertex V, while obviously not all vertices contribute significant structural information.

In this paper, we propose a modification to standard graph filtration that will resolve both of these issues and lead to more concise and informative persistence diagrams.

We will first define the filtration values for the vertices. For every vertex and value $t > 0$, we define $N_t(v)$ to be the number of vertices in the connected component of $\mathcal{G}_t^*$ that contains v.Fix $k \geq 1$, and define

$$\tau_k(v) := \inf\{t : N_t(v) \geq k\}. \tag{1}$$

The edge values are then defined as

$$\tau_k((u, v)) = \max(\tau_k(u), \tau_k(v), W((u, v))). \tag{2}$$

Denoting the corresponding filtration by $\mathcal{G}_t^{(k)}$, note that $\mathcal{G}_t^{(1)} \equiv \mathcal{G}_t^*$. In other words, compared to $\mathcal{G}_t^*$, in $\mathcal{G}_t^{(k)}$, we delay the appearance of the vertices until the first time each vertex is contained in a component with at least k vertices (and adjust the edge appearance

to be compatible). Effectively, the assignment of the new filtration values to the vertices introduces three changes to persistence diagrams:

1. All points that are linked to components smaller than k are removed;
2. Each birth time corresponds to an edge merging two components C_1, C_2 in $\mathcal{G}_t^*$, such that $|C_1|, |C_2| < k$ and $|C_1| + |C_2| \geq k$;
3. Each death time corresponds to an edge merging two components with at least k vertices each.

We call this filtration approach 'k-cluster filtration' to represent the fact that it tracks the formation and merging of clusters of at least k. The parameter k determines what we consider to be a sufficiently meaningful cluster. In $\mathcal{G}_t^*$, every vertex is considered to be a cluster but, statistically speaking, this is overkill. The chosen value of k should depend on the application, as well as the sample size.

We conclude this section by showing that k-cluster filtration decreases (in a set sense) as we increase k. This could be useful, for example, in the context of multi-parameter persistence, which we briefly mention later but leave for future work.

Lemma 1. *Filtration $\mathcal{G}_t^{(k)}$ decreases in k or equivalently:*

$$\tau_{k-1}(x) \leq \tau_k(x), \quad \forall x \in V \cup E.$$

Proof. For any vertex $v \in V$, if $|N_t(v)| \geq k$, then $|N_t(v)| \geq k-1$. From (1), we therefore have that $\tau_{k-1}(v) \leq \tau_k(v)$. Using (2), we have $\tau_{k-1}(e) \leq \tau_k(e)$ for all $e \in E$. □

4. Algorithm

In this section, we describe an efficient one-pass algorithm for computing the filtration function and persistence diagram at the same time. The time complexity of the algorithm is $O(|E| \times \alpha(|V|))$, where $\alpha(\cdot)$ is the inverse Ackermann function [16]. This is the same complexity as when computing 0-dimensional persistence diagrams if we were given the filtration function as input.

We begin with the (standard) terminology and data structures. For simplicity, we assume that the weights of the edges are unique and that the vertices have a lexicographical order. We first define the filtration function, which determines the total ordering of the vertices. Initially, undefined filtration function values are assumed to be ∞. If the function value is the same or undefined for two vertices, the order is determined via lexicographical ordering. It is then straightforward to check this is a total ordering.

Remark 1. *In the case of a total ordering, we can choose a representative 0-dimensional persistent homology class. Notably, in total ordering, a unique vertex is the earliest generator for a homology class (i.e., cluster), which we denote as a canonical representative of the persistent component.*

To track components as we proceed incrementally through the filtration approach, we use the union-find data structure, which supports two operations:

- ROOT(v) returns the canonical representative for the connected component containing v;
- MERGE(u, v) merges the connected components containing u and v into one component, including updating the root.

We augment the data structure by keeping track of two additional records:

- SIZE(v) returns the size of the connected component containing v;
- COMPONENT(v) returns the list of vertices in the same component as v.

To track the size of the component, we store the size at the root (i.e., the canonical representative) of each component, updating it each time a merge occurs. To access a connected component, recall that union-find data structures can be implemented as rooted trees. For each vertex, we store a list of children in the tree. To recover the list of vertices in

the component, we perform a depth-first search of the tree, starting from the root (although any other traversal method could be used). All update operations have $O(1)$ cost (cf., [16]).

Note that when $k = 1$, the filtration value of each vertex is 0; so, the problem reduces to finding the minimum spanning tree of a weighted graph. Hence, we assume that $k > 1$. Initially, we set the filtration functions $\tau(v) = 0$ for all vertices and $\tau(e) = W(e)$ for all edges and assume that the edges are sorted by increasing weight. Note that if this is not the case, this step will become a bottleneck, with a cost of $O(|E| \log |E|)$. Thus, we begin with a forest where each component is a single vertex, i.e., all components are initially born at 0.

We proceed as in the case of standard 0-dimensional persistence, by adding edges incrementally. As all components are present at time 0, we are only concerned with merges. The problem is reduced to updating birth times as we proceed by keeping track of 'active' components, i.e., those larger than k. We omit points in persistence diagrams that are on the diagonal (death = birth), although these may be included with some additional bookkeeping.

Assume we are adding the edge $e = (u, v)$. If e is internal to a connected component (i.e., $\text{ROOT}(u) = \text{ROOT}(v)$), then it does not affect the 0-persistence. Otherwise, it connects the two components denoted as C_u, C_v. There are a few cases to consider:

1. $|C_u \cup C_v| < k$: The merged component is too small to affect the persistence diagram, so we only perform a merge of the components;
2. $|C_u \cup C_v| \geq k$ and $|C_u| < k$: In this case, C_u becomes active. Thus, we merge the components and update the value of τ for all vertices in C_u.

$$\tau(x) \leftarrow W(e) \qquad \forall x \in C_u$$

   is performed. We take similar action if $|C_v| < k$ (or if both are less than k);
3. $|C_u|, |C_v| \geq k$: Both components are already active and so a new point $(\text{birth}, \text{death})$ is added to the persistence diagram using

$$\begin{aligned} \text{birth} &= \max\{\tau(\text{ROOT}(u)), \tau(\text{ROOT}(v))\}, \\ \text{death} &= W(e). \end{aligned}$$

   The components are again merged. We note that for any v,

$$\tau(\text{ROOT}(v)) = \min_{x \in C_v} \tau(x).$$

The full procedure is given in Algorithm 1. Note that we only compute the filtration of the vertices as the correct edge values can then be computed using Equation (2).

Proof of Correctness. We first argue that the function τ is correctly computed. This follows directly from the fact that the algorithm explicitly tests that the components contain at least k vertices. The fact that the corresponding persistence diagram is correctly computed is a consequence of the following result. □

Lemma 2. *The minimum spanning tree for $k = 1$ is the minimum spanning tree for any k.*

Proof. The key observation is that until a component contains k vertices, any spanning tree is a minimum spanning tree as all edges are assigned the value of when the component becomes active. The values of the remaining edges do not change and so remain in the MST. □

The equivalence of the MST and persistence diagram [17] then implies the correctness of the algorithm.

Proof of Running Time. The analysis of the merging is covered verbatim in the standard analysis of union-find data structures [16]. As described above, updating the sizes of the components and the lists of children in the merges are $O(1)$ operations. All that remains to prove is the cost of updating the function τ. We observe that each vertex is only updated once. This has a total cost of $O(|V|)$, while the edges can be updated at a cost of $O(1)$ each. However, if we only want to obtain a persistence diagram (and not the actual graph filtration), we do not need to update the edges since we perform a single pass. Therefore, the overall running time is $O(|E| \times \alpha(|V|))$. □

Extracting the Clusters. To obtain clusters, we can use the algorithm in [12]. This algorithm extracts the ℓ-most persistent clusters by only performing merges when the resulting persistence is less than a given threshold. This threshold can be chosen such that there are only ℓ points above the threshold in the diagram. Finally, we note that cluster extraction can be performed on an MST rather than a full graph.

Algorithm 1 The one-pass algorithm

```
G = (V, E, W)
τ : V → [0, ∞)
Initialize union-find data structure: ROOT(v) = v for all v ∈ V
Dgm, MST = ∅
for e = (u, v) ∈ E do
    if ROOT(u) ≠ ROOT(v) then
        MST ← MST ∪ e
        if SIZE(u) + SIZE(v) > k then
            if SIZE(u) < k then
                for x ∈ COMPONENT(u) do
                    τ(x) ← W(e)
                end for
            end if
            if SIZE(v) < k then
                for x ∈ COMPONENT(v) do
                    τ(x) ← W(e)
                end for
            end if
            if SIZE(u), SIZE(v) ≥ k then
                birth = max{τ(ROOT(u)), τ(ROOT(v))}
                death = W(e)
                Dgm ← Dgm ∪ (birth, death)
            end if
        end if
        MERGE(u, v)
    end if
end for
return Dgm, MST, τ
```

5. Experiments and Applications

In this section, we probe the behavior of k-cluster persistence through a sequence of experiments. Using several synthetic datasets, we study the dependence on the parameter k and experimentally show that this filtration approach follows the universality laws discovered in [13]. Next, we demonstrate how k-cluster persistence can be used for clustering by taking advantage of universality to automatically determine the number of 'statistically significant' clusters. Finally, we show how our new clustering method works in more exotic spaces, specifically tree structures. The results presented here are a preliminary investigation into this construction and we expect to continue with more applications in future work.

Remark 2. *Note that in all persistence diagrams presented in the figures, the axes are modified to birth vs. death/birth (rather than birth vs. death).*

5.1. Simulated Point Clouds

We start by generating point clouds from a mixture of Gaussians, resulting in several clusters of points (Figure 2). We first show the effect of the parameter k on the filtration function and the corresponding persistence diagrams. For the two point clouds in Figure 2, we show the resulting persistence diagrams for the k-cluster filtration functions in Figure 3. Notice that the correct number of persistent clusters is evident, especially for $k = 10, 20$, and 50. An important phenomenon that is evident in the figures is that higher values of k filter out more of the 'noise'.

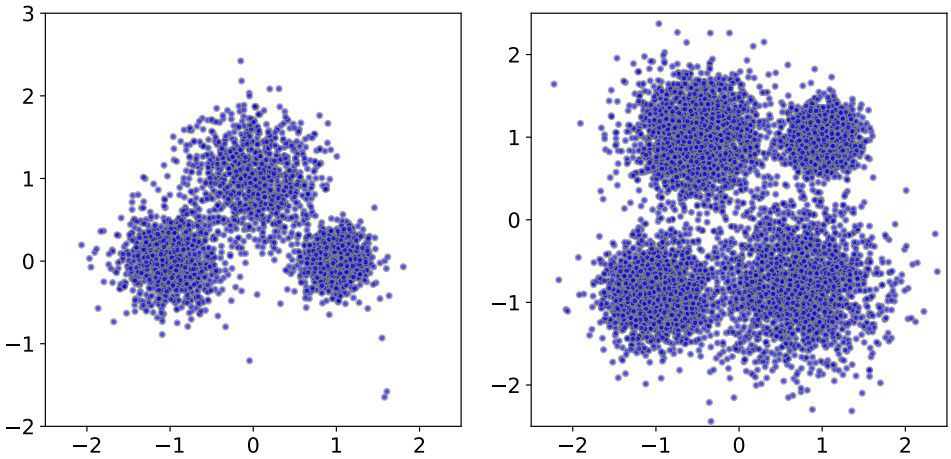

Figure 2. Two examples of point clouds consisting of IID sampling from a mixture of three and four Gaussian functions, with 1000 and 2000 points, respectively.

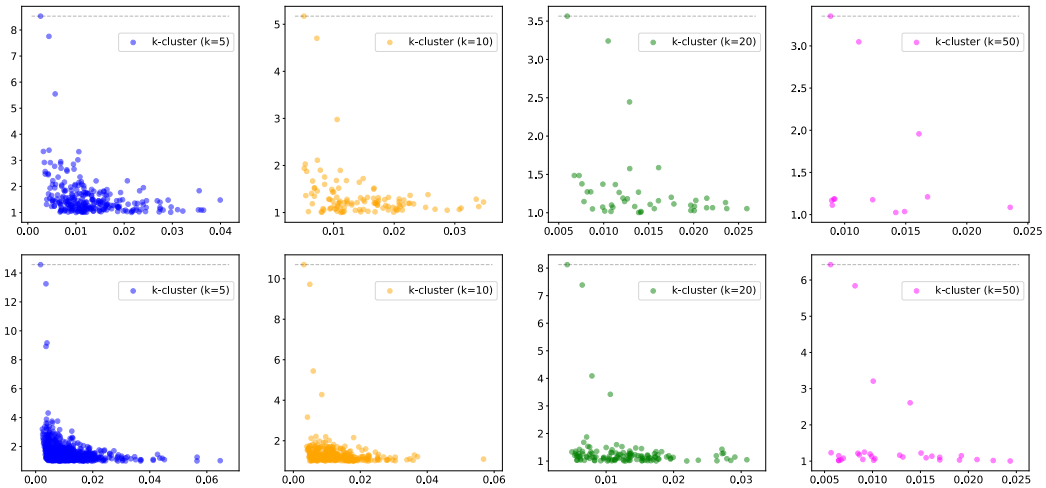

Figure 3. The persistence diagrams, with death/birth on the y-axis and different values of k for the points sampled from the two mixtures of Gaussian functions (**top** row: 3 groups; **bottom** row: 4 groups). Note that the number of outstanding features in the diagrams correspond to the number of clusters in the data.

To place the behavior of the persistence diagrams into further context, we compare the k-cluster filtration method to a related construction from the applied topology literature, which has been suggested for dealing with outliers in clustering (and higher homological dimensions): k-degree Vietoris–Rips filtration [18]. Given a weighted graph $G = (V, E, W)$, we define the k-degree filtration function $\delta_k : (V \cup E) \to [0, \infty)$ as follows: for every vertex $v \in V$, we take $\delta_k(v)$ to be its k-nearest neighbor distance. The values of the edges are then

determined as in (2). The k-degree filtration approach has been used in the context of multi-parameter persistence, with bifiltration induced by decreasing k and increasing the edge weight (commonly, Euclidean distance). In this paper, we do not explore multi-parameter settings. Rather, we focus the properties of persistence diagrams for fixed values of k. We make two observations before investigating the differences:

1. The k-degree filtration function is determined completely by the local neighborhood of a vertex, i.e., its immediate neighbors in the graph. The same is not true for k-cluster filtration;
2. For a fixed value of k, we have $\tau_k(v) \leq \delta_{k-1}(v)$ for all $v \in V$. In other words, the value of the k-cluster function is less than or equal to the value of the $(k-1)$-degree function. This follows from the fact that if a vertex has $k-1$ neighbors, then it is part of a cluster with at least k vertices.

In Figure 4, we show the persistence diagrams (i.e., birth vs. death/birth) for two non-convex clusters for both the k-degree and k-cluster filtration methods, using different values of k. In this example, especially for larger k values, the persistent clusters are much more prominent in k-cluster filtration compared to k-degree filtration. This may be explained by the fact that a much larger radius is needed to obtain the required number of neighbors. In Figure 5, we show the same comparison for persistence diagrams for 3 and 4 groups, where the difference between the two methods is less clear. However, Figure 6 highlights an additional difference between the behaviors of the two filtration approaches. In this figure, we compare persistence (death/birth) for the second most persistent cluster, using a wide range of k values. In the left and center plots, the second most persistent clusters correspond to true clusters in the data. We observe that the persistence value decays much more slowly for k-cluster filtration, i.e., the true cluster remains more persistent for increasing values of k. The plot on the right presents the same comparison but for uniformly distributed random points. In this case, the second most persistent cluster is construction noise, i.e., not a real cluster in the data. Although k-cluster filtration still decays more slowly, it is at a comparable rate to that of k-filtration. Hence, we can conclude that persistent clusters show more stable behavior over ranges of k for k-cluster filtration compared to k-degree filtration.

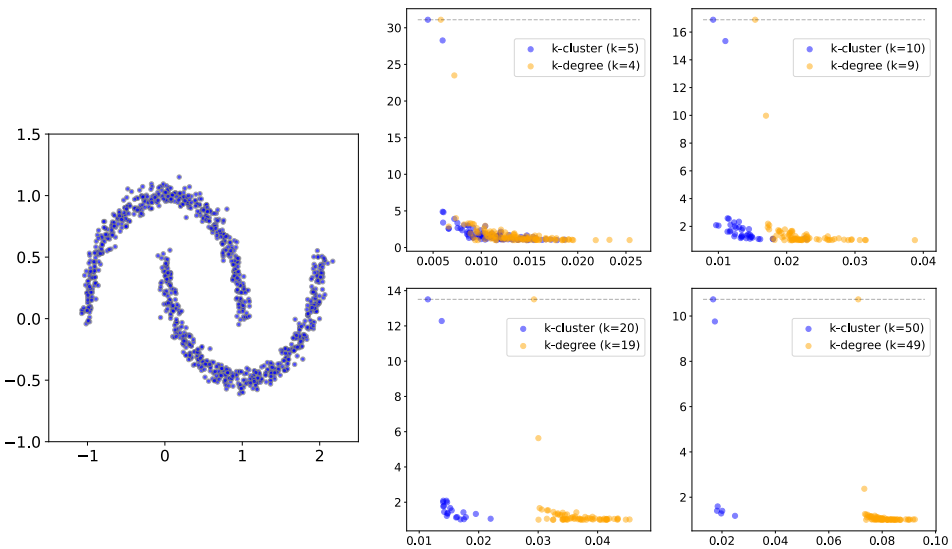

Figure 4. A comparison of k-cluster and k-degree filtrations for the two moons dataset. On the right, we have the persistence diagrams for different values of k.

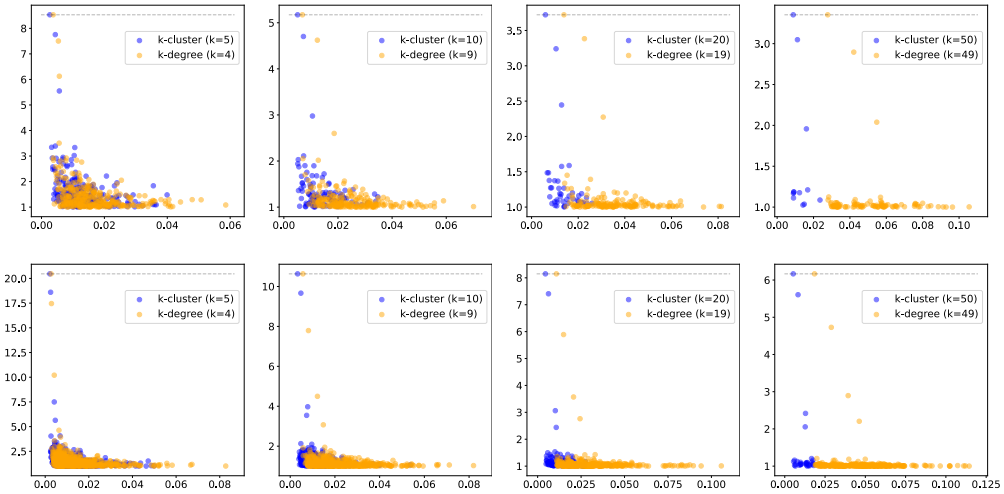

Figure 5. The persistence diagrams for each point cloud, with k-degree filtration presented in yellow and k-cluster filtration presented in blue for $k = 5, 10, 20$, and 50 (**top row**: 3 clusters; **bottom row**: 4 clusters).

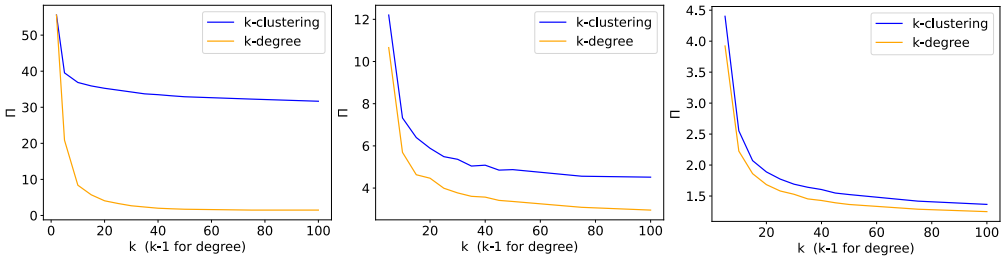

Figure 6. The effect on the second most persistent cluster for different values of k. In the left and center panels, this corresponds to true clusters (**left**: two moons; **center**: a mixture of 3 Gaussians). The right panel presents the results for uniformly random points. Here, the noise cluster drops nearly as quickly in both cases.

5.2. Universality

In [13], we published a comprehensive experimental work showing that the distribution of persistence values is universal. We consider persistence diagrams as finite collections of points in $\mathbb{R}^2$: $\mathrm{dgm} = \{(b_1, d_1), \ldots, (b_M, d_M)\}$. For each point $p_j = (b_j, d_j)$, we consider the multiplicative persistence value $\pi(p_j) = d_j/b_j$. Our goal is to study the distributions of π-values across entire diagrams.

Our results in [13] are divided into two main parts. Given a point cloud of size n, we compute the persistence diagrams for Čech and Vietoris–Rips filtrations. In *weak universality*, we consider the empirical measure

$$\Pi_n := \frac{1}{|\mathrm{dgm}_i|} \sum_{p \in \mathrm{dgm}_i} \delta_{\pi(p)},$$

and we conjecture that for IID samples, we have

$$\lim_{n \to \infty} \Pi_n = \Pi^*_{d,\mathcal{T},i},$$

where d is the dimension of the point cloud, i is the degree of homology, and $\mathcal{T}$ is the filtration type, i.e., Čech or Vietoris–Rips. In other words, the limiting distributions for the π-values depend on $d, i, \mathcal{T}$ but are independent of the probability distributions generating the point clouds.

In *strong universality*, we present a much more powerful and surprising conjecture. Here, we define $\ell(p) := A \log\log(\pi(p)) + B$ (the values of A and B are specified in [13]) and the empirical measure

$$\mathcal{L}_n := \frac{1}{|\mathrm{dgm_i}|} \sum_{p \in \mathrm{dgm}_i} \delta_{\ell(p)}.$$

Our conjecture is that for wide class of random point clouds (including non-IID and real data), we have

$$\lim_{n\to\infty} \mathcal{L} = \mathcal{L}^*,$$

where $\mathcal{L}^*$ is a unique universal limit. Furthermore, we conjecture that $\mathcal{L}^*$ might be the left-skewed Gumbel distribution.

Originally, the results in [13] were irrelevant for the 0-th persistence diagrams of random point clouds as the birth times were all zero. However, once we replace standard filtration with k-cluster filtration, we obtain new persistence diagrams with non-trivial birth times that we can study. In Figure 7, we demonstrate both weak and strong universality properties for k-cluster persistent homology. We generated IID point clouds across different dimensions and with different distributions (uniform in a box, exponential, normal). The results show that both weak and strong universality hold in these cases as well. We note that for weak universality, the limiting distribution depends on both d (the dimension of the point cloud) and k (the minimum cluster size).

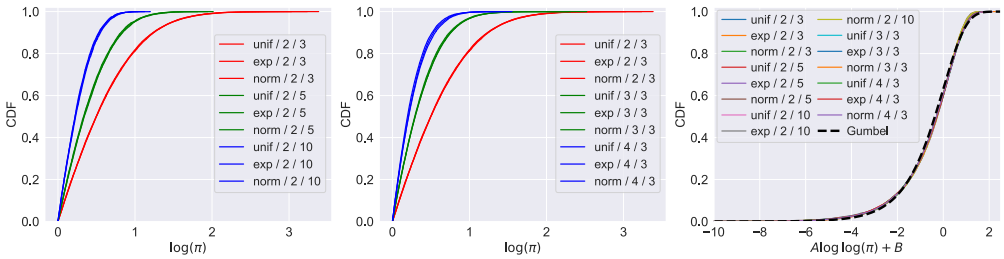

Figure 7. Universal distribution for k-cluster persistence. The labels in the legend are structured as distribution/d/k, where d is the dimension of the point cloud and k is the cluster size. The distributions taken are uniform in unit boxes, exponential, and normal. The first two plots show that weak universality holds and that the limit depends on d, k, but not on the distribution. The rightmost plot demonstrates that strong universality holds under proper normalization. We also include the left-skewed Gumbel distribution (dashed line) for comparison.

5.3. Clustering

As mentioned in the introduction, a key motivation for this work was to apply k-cluster filtration to clustering. To obtain clustering from a 0-dimensional persistence diagram, we use the algorithm proposed in [12]. Roughly speaking, given a threshold α, the algorithm extracts all clusters that are at least α-persistent. We note that the original measure for persistence in [12] was given by $d - b$; however, the change to use d/b in the algorithm is trivial.

Statistical Testing. An important consequence of the universality results presented in Section 5.2 is that the limiting distribution (after normalization) appears to be a known distribution, i.e., the left-skewed Gumbel. We can thus perform statistical testing on the number of clusters, as in [13]. The null hypothesis denoted by $\mathcal{H}_0^{(i)}$ is that the i-th most persistent cluster is due to noise. Assuming that the universality conjectures hold, the null hypothesis is given in terms of the ℓ-values as

$$\mathcal{H}_0^{(i)} \ : \ \ell(p_i) \sim \mathrm{LGumbel}.$$

where p_i represents the i-th most persistent cluster in terms of death/birth. The corresponding p-value is given by

$$p\text{-value}_i = \mathbb{P}\left(\ell(p_i) \geq x \mid \mathcal{H}_0^{(i)}\right) = e^{-e^x}.$$

Note that since we are testing sorted values, we must use multiple hypothesis testing corrections. In the experiments we describe below, we use the Bonferroni correction.

In Figure 8, we compare k-cluster filtration and k-degree filtration using persistence-based clustering from [12] and other common algorithms for clustering. For the other approaches, we use the standard implementations found in [19], which have associated techniques for choosing the number of clusters. In the cases of k-cluster filtration and k-degree filtration, the numbers of clusters are chosen using the statistical testing described above. Note that since the numbers of points in the standard examples are quite small, we limit k to 5 and 10. The best result is for k-cluster filtration with $k = 10$ ($k = 5$ fails to identify one of the clusters in the third example). On the other hand, k-degree filtration performs well but the additional 'noise' points in the diagram mean that some clusters are not identified as significant.

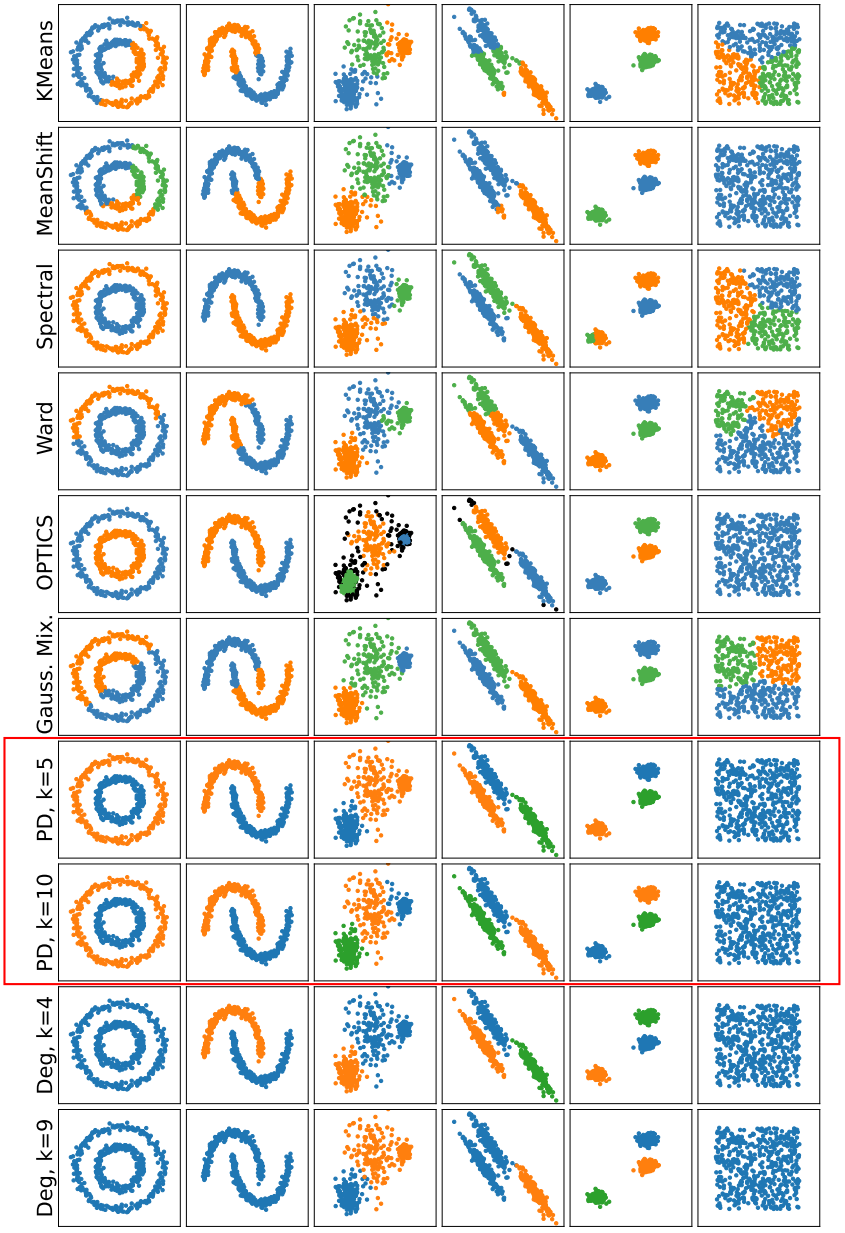

Figure 8. A comparison of standard clustering examples using different clustering approaches. In the case of k-cluster filtration (PD) and k-degree filtration (Deg), the numbers of clusters are chosen using statistical significance testing.

Clustering on Trees. As a second example, we describe clustering on weighted trees. We generate a uniform random tree on n vertices and assign uniformly distributed random weights on the edges (between 0 and 1). We show an example in Figure 9. The method seems to capture a certain structure of the tree, although we leave further investigation of this structure as future work.

Note that in the tree case, it is often impossible to use k-degree filtration as the trees have vertices with degrees that are smaller than k, which will never be included in filtration, whereas for k-cluster filtration, all nodes are included as long as the underlying graph is connected (or all components have at least k vertices). We note that it is possible to use alternative definitions for k-degree filtration by embedding the trees into metric spaces (i.e., using graph metrics induced by the weights). However, this is similar to studying complete graphs induced by the metrics, which is somewhat different from studying the graphs directly. We demonstrate this method in the rightmost plot of Figure 9.

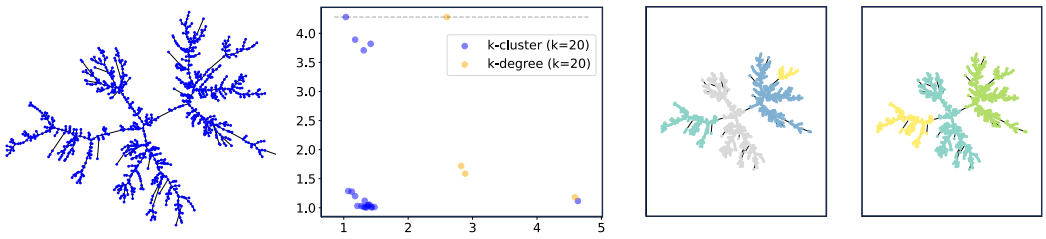

Figure 9. Clustering on a uniform random tree. The threshold gives 4 clusters for k-cluster filtration but only 3 for (metric) k-degree filtration.

6. Probabilistic Analysis

In this section, we revisit some fundamental results for random graphs and random simplicial complexes and show that analogous statements hold for our new k-cluster filtration approach. We provide the main statements, while the proofs are available in the appendix. As real data are random, we view these statements (and their future expansions) as an integral part of the analysis of k-cluster persistence, within the context of topological data analysis.

6.1. Connectivity

We consider two models here. In the $G(n,p)$ random graph, we have n vertices and each edge is placed independently with probability p. In the $G(n,r)$ random geometric graph, we take a homogeneous Poisson process $\mathcal{P}_n$ on the d-dimensional flat torus, with rate n. Edges are then placed between vertices that are less than r apart. In both models, connectivity results are tied to the expected degree. For the $G(n,p)$ model, we define $\Lambda = np$, while for the $G(n,r)$ model, we take $\Lambda = n\omega_d r^d$. Then, in [20,21], the following was proved.

Theorem 1. *Let G_n be either $G(n,p)$ or $G(n,r)$. Then*

$$\lim_{n\to\infty} \mathbb{P}(G_n \text{ is connected}) = \begin{cases} 1 & \Lambda = \log n + w(n), \\ 0 & \Lambda = \log n - w(n). \end{cases}$$

A key element in proving connectivity (for either model) is to show that around $\Lambda = \log n$, the random graph consists of a single giant component, a few isolated vertices, and nothing else. Thus, connectivity is achieved when the last isolated vertices become connected.

Our goal in this section is to analyze connectivity in the $G(n,p)$ and $G(n,r)$ models using our new k-cluster filtration method. Note that for a fixed value of n, we can view both models as filtrations over the complete graphs. For the $G(n,p)$ model, the weights of the edges are independent random variables that are uniformly distributed in $[0,1]$. For the $G(n,r)$ model, the weight of an edge is given by the distance between the corresponding points in the torus. We define $G^{(k)}(n,p)$ and $G^{(k)}(n,r)$ as the random filtrations generated

by changing the filtration function to τ_k. Our goal here is to explore the phase transitions for k-cluster connectivity. As opposed to connectivity in the original random graphs, these results differ between the models.

Theorem 2. *For the $G^{(k)}(n,p)$ filtered graph, we have*

$$\lim_{n\to\infty}\mathbb{P}\Big(G^{(k)}(n,p) \text{ is connected}\Big) = \begin{cases} 1 & \Lambda = \frac{1}{k}(\log n + (k-1)\log\log n) + w(n), \\ 0 & \Lambda = \frac{1}{k}(\log n + (k-1)\log\log n) - w(n), \end{cases}$$

for any $w(n) = o(\log\log n)$, such that $w(n) \to \infty$.

For the $G^{(k)}(n,r)$ model, proving the connectivity is a much more challenging task and beyond the scope of this paper. However, the following statement is relatively straightforward to prove.

Proposition 1. *Let $N_k = N_k(n,r)$ be the number of connected components of size k in $G(n,r)$. Then,*

$$\lim_{n\to\infty}\mathbb{P}(N_k = 0) = \begin{cases} 1 & \Lambda = \log n - (d-1)(k-1)\log\log n + w(n), \\ 0 & \Lambda = \log n - (d-1)(k-1)\log\log n - w(n). \end{cases}$$

for any $w(n) = o(\log\log n)$, such that $w(n) \to \infty$.

From this lemma, we conclude that when $\Lambda = \log n - (d-1)(k-1)\log\log n - w(n)$, the graph $G(n,r)$ has components of size k, which implies that $G^{(k)}(n,r)$ is not connected. On the other hand, when $\Lambda = \log n - (d-1)(k-1)\log\log n + w(n)$, we have $N_j = 0$ for all fixed $j \geq k$, which indicates that $G^{(k)}(n,r)$ should be connected. This leads to the following conjecture.

Conjecture 3. *For the $G^{(k)}(n,r)$ filtered graph, we have*

$$\lim_{n\to\infty}\mathbb{P}\Big(G^{(k)}(n,r) \text{ is connected}\Big) = \begin{cases} 1 & \Lambda = \log n - (d-1)(k-1)\log\log n + w(n), \\ 0 & \Lambda = \log n - (d-1)(k-1)\log\log n - w(n). \end{cases}$$

Note that both phase transitions occur before those for the original graph models. This is due to the fact that for $k > 1$, k-cluster filtration does not allow any isolated vertices. Also note that when taking $k = 1$, both results coincide with Theorem 1.

6.2. Limiting Persistence Diagrams

In [22], it is shown that for stationary point processes, persistence diagrams have non-random limits (in the vague convergences of measures). A similar statement holds for k-cluster persistence diagrams.

Let $\mathrm{Dgm}^{(k)}(\mathcal{P})$ be the k-cluster persistence diagram for point cloud $\mathcal{P}$. We define the discrete measure on $\mathbb{R}^2$ as

$$\xi^{(k)}(\mathcal{P}) := \sum_{(b,d)\in \mathrm{Dgm}^{(k)}(\mathcal{P})} \delta_{(b,d)}.$$

Let $Q_L = [-L/2, L/2]^d$. The following is an analog of Theorem 1.5 in [22].

Theorem 4. *Assume that $\mathcal{P}$ is a stationary point process in $\mathbb{R}^d$, with all finite moments. For any k, there exists a deterministic measure μ_k, such that*

$$\lim_{L\to\infty}\frac{1}{L^d}\mathbb{E}\Big\{\xi^{(k)}(\mathcal{P}\cap Q_L)\Big\} = \mu_k,$$

where the limit is in the sense of vague convergence. Furthermore, if $\mathcal{P}$ is ergodic, then almost surely

$$\lim_{L\to\infty} \frac{1}{L^d} \xi^{(k)}(\mathcal{P} \cap Q_L) = \mu_k.$$

6.3. Maximal Cycles

In [23], the largest cycles in persistence diagrams are studied. In particular, the behavior of the largest π-value: $\pi(p) = d/b$ arising from a homogeneous Poisson process $\mathcal{P}_n$ is studied. Let $\Pi_{i,\max}$ be the largest π-value in the i-th persistent homology. The main result in [23] then states that, with high probability,

$$A_i \Delta_i(n) \leq \Pi_{i,\max} \leq B_i \Delta_i(n),$$

where $A_i, B_i > 0$ are constants, and

$$\Delta_i(n) = \left(\frac{\log n}{\log\log n}\right)^{1/i}.$$

For k-cluster persistence, we show that the largest π-value has a completely different scaling.

Theorem 5. *Let $\mathcal{P}_n$ be a homogeneous Poisson process in the flat torus, with rate n. Let $\Pi^{(k)}_{\max}$ denote the maximum π-value in the k-cluster persistence diagram (excluding the infinite cluster). Then, for every $\epsilon > 0$, we have*

$$\lim_{n\to\infty} \mathbb{P}\left(n^{\frac{1}{d(k-1)}-\epsilon} \leq \Pi^{(k)}_{\max} \leq n^{\frac{1}{d(k-1)}+\epsilon} \right) = 1.$$

Remark 3. *We observe that the largest π-value in k-cluster persistence is significantly larger than that in i-dimensional homology. The main reason for this can be explained as follows. In [23], our upper bound for $\Pi_{i,\max}$ is an iso-perimetric inequality, which implies that large π-values require large connected components. However, the π-values in k-cluster persistence only require clusters of size k to be formed; thus, they can be generated by much smaller connected components.*

Author Contributions: Conceptualization, O.B. and P.S.; methodology, O.B. and P.S.; software, O.B. and P.S.; writing—original draft, O.B. and P.S.; writing—review and editing, O.B. and P.S.; visualization, O.B. and P.S. All authors have read and agreed to the published version of the manuscript.

Funding: O.B. was partially supported by the Israel Science Foundation grant #1965/19. P.S. was partially supported by the EU project EnRichMyData (GA 101070284).

Data Availability Statement: No new data were created or analyzed in this study. Data sharing is not applicable to this article.

Conflicts of Interest: The authors declare no conflict of interest.

Appendix A. Connectivity

Proof of Theorem 2. Note that for k-cluster filtration, connectivity is equivalent to the original $G(n,p)$ graph having no components of size j for any $k \leq j \leq n/2$. Let $N_j = N_j(n,p)$ be the number of components of size j in $G(n,p)$. Taking similar steps to the proof of connectivity for random graphs (e.g., [24]), we have

$$\mathbb{E}\{N_j\} \leq \binom{n}{j} j^{j-2} p^{j-1} (1-p)^{j(n-j)}. \tag{A1}$$

For $k+1 \leq j < 4k$, we have

$$\mathbb{E}\{N_j\} \leq Cn^j \left(\frac{\Lambda}{n}\right)^{j-1} e^{-j(n-4k)(\Lambda/n)},$$

for some $C > 0$. Taking $\Lambda = \frac{1}{k}(\log n + (k-1)\log\log n) + c$, we have

$$\mathbb{E}\{N_j\} \leq Cn^{-1/k}(\log n)^{j-1} e^{4j\log n/n}.$$

For $4k \leq j \leq n/2$, we have

$$\mathbb{E}\{N_j\} \leq \left(\frac{ne}{j}\right)^j j^{j-2}\left(\frac{\Lambda}{n}\right)^{j-1} e^{-j\Lambda/2} \leq \frac{n}{j^2}\left(\frac{e^{1-c/2}\log n}{n^{1/2k}}\right)^j \leq \frac{n}{j^2} n^{-j/3k}.$$

Therefore,

$$\sum_{j=4k}^{n/2} \mathbb{E}\{N_j\} \leq \frac{1}{8k^2} n^{-1/3}.$$

To conclude, we show that

$$\lim_{n\to\infty} \sum_{j=k+1}^{n/2} \mathbb{E}\{N_j\} = 0.$$

This implies that for $\Lambda = \frac{1}{k}(\log n + (k-1)\log\log n) + c$, we have

$$\mathbb{P}\Big(G^{(k)}(n,p) \text{ is connected}\Big) \approx \mathbb{P}(N_k > 0).$$

Similar estimates to those above show that

$$\mathbb{E}\{N_k\} \approx e^{-kc}.$$

Therefore, when $c = w(n) \to \infty$, we have $\mathbb{P}(N_k > 0) \to 0$. Together with a second-order argument, we can similarly show that when $c = -w(n)$, we have $\mathbb{P}(N_k > 0) \to 1$. This concludes the proof. □

Proof of Proposition 1. Recall that N_k is the number of components of size k in $G(n,r)$. In [25] (Theorem 3.3), it is shown that

$$\mathbb{E}\{N_k\} \approx \mathrm{Var}(N_k) \approx C_k n \Lambda^{-(d-1)(k-1)} e^{-\Lambda},$$

for some constant $C_k > 0$. When $\Lambda = \log n - (d-1)(k-1)\log\log n + w(n)$, we have $\mathbb{E}\{N_k\} \to 0$, implying that $\mathbb{P}(N_k > 0) \to 0$. When $\Lambda = \log n - (d-1)(k-1)\log\log n - w(n)$, we can use Chebyshev's inequality:

$$\mathbb{P}(N_k = 0) \leq \mathbb{P}(|N_k - \mathbb{E}\{N_k\}| \geq \mathbb{E}\{N_k\}) \leq \frac{\mathrm{Var}(N_k)}{(\mathbb{E}\{N_k\})^2} \approx \frac{1}{\mathbb{E}\{N_k\}} \to 0.$$

This completes the proof. □

Appendix B. Maximal π-Value

Proof. Let r, R denote the birth and death radii of a cluster in a k-cluster persistence diagram and recall that $\pi = R/r$.

For an upper bound, we denote the number of connected subsets of size k at radius r as $N_k(r)$. Using Mecke's formula (cf. [26]),

$$\begin{aligned}\mathbb{E}\{N_k(r)\} &= \frac{n^k}{k!}\int_{(\mathbb{T}^d)^k} \mathbb{1}\{G(\mathbf{x},r) \text{ is connected}\}d\mathbf{x} \\ &= \frac{n\lambda^{k-1}}{k!}\int_{(\mathbb{R}^d)^{k-1}} \mathbb{1}\{G((0,\mathbf{y}),1) \text{ is connected}\}d\mathbf{y}, \\ &= C_k n\lambda^{k-1},\end{aligned} \tag{A2}$$

where C_k is a positive constant, $\lambda = nr^d$, and we use the change in variables $x_i \to x_1 + ry_i$ $(i = 2,\ldots,k)$. For any $\epsilon > 0$, if $\lambda = n^{-1/(k-1)-\epsilon}$, then $\mathbb{E}\{N_k(r)\} \to 0$. Thus, we can assume with high probability that the birth times of all k-clusters have $\lambda \geq n^{-1/(k-1)-\epsilon}$. In addition, from Theorem 1, if we denote $\Lambda = nR^d$, then when $\Lambda = C\log n$, the graph $G(n,r)$ is connected. This implies with high probability that all death times of k-clusters have $\Lambda \leq C\log n$.

Together, these bounds imply with high probability that for all points in k-cluster persistence diagrams, for any $\epsilon > 0$, we have

$$\pi = \left(\frac{\Lambda}{\lambda}\right)^{1/d} \leq \left(\frac{C\log n}{n^{-1/(k-1)-\epsilon}}\right)^{1/d}.$$

Therefore, for any $\epsilon > 0$, we have

$$\lim_{n\to\infty} \mathbb{P}\left(\Pi^{(k)}_{\max} \leq n^{\frac{1}{d(k-1)}+\epsilon}\right) = 1.$$

For the lower bound, we denote the number of components of size k that are born before r and are isolated at radius R (and hence die after R) as $\hat{N}_k(r,R)$. Then,

$$\begin{aligned}\mathbb{E}\{\hat{N}_k(r,R)\} &= \frac{n^k}{k!}\int_{(\mathbb{T}^d)^k} \mathbb{1}\{G(\mathbf{x},r) \text{ is connected}\}e^{-n\,\mathrm{Vol}(B_R(\mathbf{x}))}d\mathbf{x}, \\ &= \frac{n\lambda^{k-1}}{k!}\int_{(\mathbb{R}^d)^{k-1}} \mathbb{1}\{G((0,\mathbf{y}),1) \text{ is connected}\}e^{-n\,\mathrm{Vol}(B_R(0,r\mathbf{y}))}d\mathbf{y},\end{aligned}$$

where $B_R(\mathbf{x})$ is the union of balls of radius R around $\mathbf{x}$. We apply the dominated convergence theorem, using the fact that when $r/R \to 0$, we have

$$\lim_{n\to\infty} \frac{\mathrm{Vol}(B_R(0,0+r\mathbf{y}))}{\omega_d R^d} = 1.$$

This leads to

$$\mathbb{E}\{\hat{N}_k(r,R)\} \approx C_k n\lambda^{k-1}e^{-\omega_d\Lambda}.$$

Taking $\Lambda = C > 0$ and $\lambda = n^{-1/(k-1)+\epsilon}$, we have

$$\mathbb{E}\{\hat{N}_k(r,R)\} \to \infty.$$

Using a second moment argument shows that for all $\epsilon > 0$

$$\mathbb{P}\left(\Pi^{(k)}_{\max} \geq n^{\frac{1}{d(k-1)}-\epsilon}\right) \to 1,$$

completing the proof. □

Appendix C. Limiting Persistence Diagram

The key part of the proof in [22] is bounding the add-one costs of the *persistent* Betti numbers. Let $G = (V, E, W)$ be a weighted graph and let $\{G_t^{(k)}\}$ be the corresponding k-cluster filtration function. Define $\beta_0^{r,s}(G^{(k)})$ as the 0-th persistent Betti number, i.e., the number of components born in $t \in [0, r]$ that die at $(s, \infty]$ (for a formal definition, see [22]). Fix edge $e_0 \notin E$ with a given weight $W(e_0) = w_0$ and let $\tilde{G} = (V, \tilde{E}, \tilde{W})$ be a weighted graph with $\tilde{E} = E \cup \{e_0\}$. Then,

$$\tilde{W}(e) := \begin{cases} W(e) & e \neq e_0, \\ w_0 & e = e_0. \end{cases}$$

Let $\{\tilde{G}_t^{(k)}\}$ denote the corresponding k-cluster filtration function. The entire proof of Theorem 4 follows verbatim from the proofs in [22], provided that we prove the following lemma.

Lemma A1.

$$\left|\beta_0^{r,s}(\tilde{G}^{(k)}) - \beta_0^{r,s}(G^{(k)})\right| \leq 1.$$

In other words, if we add a single edge to the filtration function, the number of persistent clusters can change by at most 1. Note that the proof here is not a straightforward application of Lemma 2.10 in [22] since in our case, when a single edge is added to the filtration function, the filtration values of other vertices and edges might be affected.

Proof. Let $e_0 = (u, v)$ with $W(e_0) = w_0$. Let C_u and C_v denote the components of the end points of e_0 at w_0 in the original filtration function $\{G_t^{(k)}\}$. There are then three possible cases that can occur.
Case I: Both $|C_u| < k$ and $|C_v| < k$. Note that in this case, $\tau_k(u), \tau_k(v) > w_0$. Let C'_u be the cluster of u at $\tau_k(u)$, so that it is the component of u when it first appears in $G_t^{(k)}$. Define C'_v similarly. Note that aside from $C'_u \cup C'_v$, the filtration values of all other vertices remain unchanged by adding e_0.

Without loss of generality, suppose that $w_0 < \tau_k(u) < \tau_k(v)$. Then, comparing the persistence diagrams for $G_t^{(k)}$ and $\tilde{G}_t^{(k)}$, only two differences can occur:

1. The point representing C'_v in $G_t^{(k)}$ is removed since C'_v is no longer a connected component in $\tilde{G}_t^{(k)}$ (as it is merged with C'_u);
2. The point representing C'_u in $G_t^{(k)}$ may show an earlier birth time in $\tilde{G}_t^{(k)}$, within the interval $[w_0, \tau_k(u))$.

For a given r, s, the first change might decrease $\beta_k^{r,s}$ by 1, while the second change might increase it by 1. In any case, the total difference between $\beta_0^{r,s}(\tilde{G}^{(k)})$ and $\beta_0^{r,s}(G^{(k)})$ is no more than 1.
Case II: $|C_u| \geq k$ and $|C_v| < k$. Defining C'_v in the same way as above, note that in this case, the filtration values of all points outside C'_v remain unchanged by adding e_0. The only change that occurs in this case is that the point in the diagram of $G_t^{(k)}$ corresponding to C'_v is removed in $\tilde{G}_t^{(k)}$ since it is now merged with C_u. Therefore, the difference in the persistent Betti numbers is at most 1.
Case III: Both $|C_u| \geq k$ and $|C_v| \geq k$. If $C_u = C_v$, then adding e_0 creates a 1-cycle (loop) and does not affect the k-cluster persistence diagram. If $C_u \neq C_v$, then both C_u and C_v are represented by different points in the persistence diagram of $G_t^{(k)}$. Adding e_0 causes one of these components to die earlier. In this case, $\beta_0^{r,s}$ may be decreased by 1 (if $s > w_0$). □

References

1. Jain, A.K.; Murty, M.N.; Flynn, P.J. Data clustering: A review. *ACM Comput. Surv.* **1999**, *31*, 264–323. [CrossRef]
2. McInnes, L.; Healy, J.; Melville, J. Umap: Uniform manifold approximation and projection for dimension reduction. *arXiv* **2018**, arXiv:1802.03426.
3. Edelsbrunner, H.; Harer, J.L. *Computational Topology: An Introduction*; American Mathematical Society: Providence, RI, USA, 2010.
4. Zomorodian, A.J. *Topology for Computing*; Cambridge University Press: Cambridge, UK, 2005; Volume 16.
5. Carlsson, G. Topology and data. *Bull. Am. Math. Soc.* **2009**, *46*, 255–308. [CrossRef]
6. Bianconi, G. *Higher-Order Networks*; Cambridge University Press: Cambridge, UK, 2021.
7. Horak, D.; Maletić, S.; Rajković, M. Persistent homology of complex networks. *J. Stat. Mech. Theory Exp.* **2009**, *2009.03*, P03034. [CrossRef]
8. Ghrist, R. Barcodes: The persistent topology of data. *Bull. Am. Math. Soc.* **2008**, *45*, 61–75. [CrossRef]
9. Carlsson, G.; Mémoli, F. Classifying clustering schemes. *Found. Comput. Math.* **2013**, *13*, 221–252. [CrossRef]
10. Carlsson, G.E.; Mémoli, F. Characterization, stability and convergence of hierarchical clustering methods. *J. Mach. Learn. Res.* **2010**, *11*, 1425–1470.
11. Bobrowski, O.; Mukherjee, S.; Taylor, J.E. Topological consistency via kernel estimation. *Bernoulli* **2017**, *23*, 288–328. [CrossRef]
12. Chazal, F.; Guibas, L.J.; Oudot, S.Y.; Skraba, P. Persistence-based clustering in Riemannian manifolds. *J. ACM* **2013**, *60*, 41. [CrossRef]
13. Bobrowski, O.; Skraba, P. A universal null-distribution for topological data analysis. *Sci. Rep.* **2023**, *13*, 12274. [CrossRef] [PubMed]
14. Hatcher, A. *Algebraic Topology*; Cambridge University Press: Cambridge, UK, 2002.
15. Munkres, J.R. *Elements of Algebraic Topology*; Addison-Wesley Reading: Reading, MA, USA, 1984; Volume 2.
16. Cormen, T.H.; Leiserson, C.E.; Rivest, R.L.; Stein, C. *Introduction to Algorithms*; MIT Press: Cambridge, MA, USA, 2022.
17. Skraba, P.; Thoppe, G.; Yogeshwaran, D. Randomly Weighted d-complexes: Minimal Spanning Acycles and Persistence Diagrams. *Comb. J. Comb.* **2020**, *27*. [CrossRef] [PubMed]
18. Lesnick, M.; Wright, M. Interactive visualization of 2-D persistence modules. *arXiv* **2015**, arXiv:1512.00180.
19. Pedregosa, F.; Varoquaux, G.; Gramfort, A.; Michel, V.; Thirion, B.; Grisel, O.; Blondel, M.; Prettenhofer, P.; Weiss, R.; Dubourg, V. Scikit-learn: Machine learning in Python. *J. Mach. Learn. Res.* **2011**, *12*, 2825–2830.
20. Erdős, P.; Rényi, A. On random graphs. *Publ. Math. Debr.* **1959**, *6*, 290–297. [CrossRef]
21. Penrose, M.D. The longest edge of the random minimal spanning tree. *Ann. Appl. Probab.* **1997**, *7*, 340–361. [CrossRef]
22. Hiraoka, Y.; Shirai, T.; Trinh, K.D. Limit theorems for persistence diagrams. *Ann. Appl. Probab.* **2018**, *28*, 2740–2780. [CrossRef]
23. Bobrowski, O.; Kahle, M.; Skraba, P. Maximally persistent cycles in random geometric complexes. *Ann. Appl. Probab.* **2017**, *27*, 2032–2060. [CrossRef]
24. Frieze, A.; Karoński, M. *Introduction to Random Graphs*; Cambridge University Press: Camrbidge, UK, 2016.
25. Penrose, M.D.; Yang, X. On k-clusters of high-intensity random geometric graphs. *arXiv* **2022**, arXiv:2209.14758.
26. Penrose, M. *Random Geometric Graphs*; Oxford University Press: Oxford, UK, 2003; Volume 5.