Fast Computation of Persistent Homology with Data Reduction and Data Partitioning

Nicholas O. Malott and Philip A. Wilsey
Dept. of EECS, University of Cincinnati, Cincinnati, OH 45221, USA
Email: malottno@mail.uc.edu, philip.wilsey@uc.edu

***Abstract*—Persistent homology is a method of data analysis that is based in the mathematical field of topology. Unfortunately, the run-time and memory complexities associated with computing persistent homology inhibit general use for the analysis of big data. For example, the best tools currently available to compute persistent homology can process only a few thousand data points in $\mathbb{R}^3$. Several studies have proposed using sampling or data reduction methods to attack this limit. While these approaches enable the computation of persistent homology on much larger data sets, the methods are approximate. Furthermore, while they largely preserve the results of large topological features, they generally miss reporting information about the small topological features that are present in the data set. While this abstraction is useful in many cases, there are data analysis needs where the smaller features are also significant (*e.g.*, brain artery analysis). This paper explores a combination of data reduction and data partitioning to compute persistent homology on big data that enables the identification of both large and small topological features from the input data set. To reduce the approximation errors that typically accompany data reduction for persistent homology, the described method also includes a mechanism of "upscaling" the data circumscribing the large topological features that are computed from the sampled data. The designed experimental method provides significant results for improving the scale at which persistent homology can be performed.**

***Index Terms*—topological data analysis; persistent homology; data reduction; data partitioning; data mining; unsupervised learning**

I. Introduction

We live in the age of data. Everyday, massive volumes of data are analyzed to extract meaningful information. This task is generally referred to as data analysis or data mining. Data analysis has grown over the past few decades to be a vast and interdisciplinary field of study encompassing statistics, mathematics, and computer science. Numerous methods have been developed to analyze large and complex data sets to extract useful knowledge. An emerging method of data analysis is based in the mathematical field of topology. Topology is the study of the properties of space that are preserved under certain types of deformations [1]. Over the last 15 years, substantial efforts have been put together to use topological methods for solving problems related to large and complicated data sets. This gave birth to a field of study called *Topological Data Analysis (TDA)* [2]–[6]. The fundamental idea is that topological methods can be used to study patterns or shapes that are preserved despite the presence of noise and variations in the data. The ability of TDA to identify shapes under certain deformations renders it immune to noise and leads to discovering properties of data that are not discernible by conventional methods of data analysis [3], [4].

The computation of *Persistent Homology* (PH) is one of the principal components in TDA. Unfortunately, the computation of PH is exponential in both time and space [6]. The result is that TDA cannot be directly applied to point clouds containing more than a few thousand data points in $\mathbb{R}^3$. One direction that has been explored to address this limitation is the application of data reduction or dimensional reduction to enable the computation of PH on large data sets [7]–[11]. While these methods enable the computation of PH on data sets that are 3-4 orders of magnitude larger, this enablement comes at a cost. In particular, the reduction of the data disables the identification of the smaller topological features in the data set as well as providing only an approximation of the specific boundaries surrounding the larger topological features. While the approximation of the large features is often sufficient for some analysis requirements, there are application areas for which the loss of the smaller topological features is a significant problem [12], [13].

This paper provides a solution to computing PH on large data sets while enabling the identification of both large and small topological features in the data. The solution uses k-means++ clustering to organize the data into partitions (the clusters). The clusters and their centroids are then independently analyzed using PH to locate the large and small topological features in the entire data set. More specifically, the data is first organized into k clusters; each cluster is then independently analyzed with a PH computation (locating the small topological features); the cluster centroids are also analyzed with a PH computation (locating the large topological features); and finally the results are merged together with duplicate results filtered. In addition, a data *upscaling* step can also be performed on the large topological features identified from the cluster centoid data to further refine their boundary computation. The actual algorithm is a slightly more complex than this and is more fully described in Section IV.

Computing persistent homology using partitioning, data reduction, and upscaling enables the application of PH to large data sets; it provides accurate identification of the large and small topological features in the data and accelerates the computation of PH by several orders of magnitude. This paper includes a systematic experimental analysis of the approach

Support for this work was provided in part by the National Science Foundation under grants ACI–1440420 and IIS–1909096.

and demonstrates the accuracy of the computation against existing PH libraries on smaller data sets that are analyzable without reduction. Experiments are also performed with large data sets that cannot be fully processed by existing tools. The results show that the approach of this paper provides accurate computations of the PH output and that it does so with runtimes that are generally 3 orders of magnitude faster than performing the PH on the entire data set.

The remainder of this paper is organized as follows. Section II presents some of the background on persistent homology. Section III summarizes the related work in this area. Section IV provides the theoretical overview of the application of partitioning and data reduction to enable and accelerate the computation of persistent homology on big data. Section V describes the implementation and limitations of the general solution outlined in Section IV that is used in the experimental testing of this paper. These approximations to the general solution are required to use existing tools for testing instead of rebuilding the entire PH infrastructure needed to evaluate this approach. Section VI presents the experimental results on several different data sets. Section VII discusses some shortcomings of the upscaling step. Finally, we conclude the paper with some remarks in Section VIII.

II. Background

This section provides a brief overview of persistent homology, its computation, and the current tools for computing the PH of a point cloud. Brief statements on the computational growth of computing PH are also given. A more detailed presentation on the fundamentals of topology are available [1] along with TDA [14], [15]. A tutorial article by Chazal is available here [2]. Otter *et al* provide a general review of the current approaches and solutions to computing PH [6].

TDA, and specifically PH, supports data analysis by identifying the topological features embedded in point clouds over different spatial resolutions. The persistence of a topological feature is observed and measured across a range of connectivity distances (called ϵ distances). Topological features identified that persist over larger intervals are generally considered significant, as they represent topological groups that exist at many different spatial resolutions. Shorter intervals have often been regarded as noise in the aspect of homology, but encode additional information about the connectedness of simplices identified in the point cloud.

The computation of PH is accomplished from simplices organized into a simplicial complex. A simplex is a generalized representation of a triangle into any number of dimensions. In this case a 0-simplex represents a point, a 1-simplex is a line, a 2-simplex is a triangle, a 3-simplex is a tetrahedron, and so on. The simplicial complex stores all simplices generated from the point cloud. In the simplest terms PH proceeds by constructing simplicial complexes at different ϵ distances (ϵ_{min} to ϵ_{max}) to create a filtration of complexes. PH then records the ϵ distances when topological features (holes, loops, and voids) appear and disappear from the filtration. The ϵ distance when a topological feature appears is called its *birth* and the ϵ distance when that feature disappears is called its *death*. The $< birth, death >$ interval then defines the persistence interval of the topological feature and can be represented in a number of different ways, namely: barcodes, persistence diagrams, persistence landscapes, and persistence images [6].

Unfortunately, the computation of persistent homology can require significant memory resources [6]. This is due to the size of the complex which grows exponentially based on the number of point cloud vectors (n), the dimension of the point cloud (d), the maximum dimension of features to compute (d_{max})[1], and minimum/maximum ϵ distance to be observed (which is generally a user defined range). This exponential growth occurs with all of these parameters and characteristics of the point cloud, consequentially growing quickly beyond millions of simplices to store and process.

The resource requirements make computing PH on current hardware infeasible for more than only a few thousand points in $\mathbb{R}^3$. Even when the complex can be constructed and evaluated the runtime for the computation can take hours due to memory access and processing. This limits the primary use of PH to relatively small, static datasets, although efforts to compute PH on large and streaming data are being pursued.

Several libraries are available to compute persistent homology; some of the recent include Ripser [16], GUDHI [17], and Eirene [18]. Ripser is a C++ library designed to compute Vietoris–Rips (VR) complexes in a memory efficient manner. GUDHI is a large C++ library for topological data analysis supporting many options of (i) complex types, (ii) complex reduction algorithms, and (iii) data analysis to support a larger understanding of TDA. Eirene is a newer project built in Julia that enables persistent homology computation and includes several other valuable analysis tools.

All three tools suffer from similar limitations in maximum data-set size as explored in Section VI. By limiting the maximum epsilon, ϵ_{max}, used to filter the complex along with the maximum dimension, d_{max}, to identify topological features in, some larger data-sets can be analyzed but still take significant time to process.

In general, point clouds have different characteristics that can cause settings on one dataset to not be applicable on another. Typically persistent homology requires careful selection of the ϵ_{max} parameter to control how large of a complex is generated from the point cloud. This limitation leaves out any features that are born, or die, after ϵ_{max} is reached and potentially discards valuable information in higher dimensions.

Computing the full PH for a large data set with high dimensional features is very challenging for current tools. As the upper bound on the range of ϵ values increase, more points become connected forming both 1-dimensional and higher dimensional simplices. This growth with higher dimensional features requires additional simplices be evaluated during the insertion and boundary reduction steps for PH. The scale of growth of the complex for big data quickly grows beyond

[1]Most PH tools allow for the computation of the homology at a limited range of dimensions to look for topological features.

limitations of system resources. Currently, PH computation on big data is infeasible without some approximation or predefined bounds of analysis.

III. Related Work

Efforts to reduce the complexity and resource requirements for computing PH on large point clouds include analysis of (i) storage, (ii) processing, (iii) approximation, and (iv) subsampling and data reduction optimizations. Storage of complexes such as the Simplex Tree [19] use memory more efficiently, leading to the successful computation of PH on larger complexes. Processing optimizations, such as the twist algorithm [20], clearing [21], and coreduction [22], have significantly improved the boundary matrix reduction step, leading to large performance increases in Ripser [16], GUDHI [17], and Eirene [18]. Approximation of PH through simplicial batch collapse has been studied by Dey *et al*, and implemented in the SimBa library [23]. Since they are closely related to the topic of this paper, additional details on studies to sample or reduce the point cloud are discussed below.

With computational costs of PH limiting the use in larger data-sets, a natural approach is to look at approximation of the simplicial complex and persistence intervals. Two major proposals exist: (i) manipulating the point cloud prior to building the simplicial complex or (ii) reducing the simplicial complex by removing non-critical simplices.

Subsampling [7], [10] and the effects of data reduction [8] on PH have been studied and successfully used to locate large topological features in a data set. Even when using random sampling, these studies show remarkable success in locating the large topological features in a large data set. Finally, the application of random projection to reduce the dimensionality of the original data set has been studied by [9], [11].

Strong collapses have provided approximation of larger features in a point cloud by reducing the number of simplices represented in the complex [24]. Cycles of the the shortest length representing a generator of a hole are preserved, enabling identification of features in large data to be attained. SimBa [23] uses simplicial batch collapse to subsample and collapse points to nearby points. Once the data is reduced, the VR-complex is built over the subsamples, providing an approximation of the complex that PH can be computed over. The resulting persistence intervals are approximations, but have shown consistent identification of large features in high dimensions with acceptable resources and runtime.

A similar approach of approximating the PH of big data point clouds was studied by Chazal *et al* [7]. Data was sampled from source point clouds and PH was computed on multiple independent samplings. Moitra *et al* [8] introduced a more targeted sampling approach by replacing the point cloud with nanoclusters — centroids of clusters identified with the k-means++ algorithm. Replacement of points in the point cloud with centroids preserved large topological features and significantly reduced the resource requirements for computation.

Experimental results with the k-means++ algorithm for replacement of the original point cloud with the computed centroids preserves the general shape of the point cloud data, even to 90% reduction of the original point cloud. Figure 1 shows this reduction property of k-means++ on a triangulated mesh point cloud of a Camel. The reduced point cloud retains the shape of the source data while thinning out dense regions around the legs, neck, and head of the camel.

Performance of the k-means++ algorithm as a preprocessor to PH shows promising results as well — in some cases as much as 3-orders of magnitude faster than without preprocessing [8]. The reduction in total vectors greatly decreases the number of simplexes stored and also thins the dense areas of the point cloud that have many connections beginning at low epsilon values. The combination of reduction and thinning provides significant results for approximating PH.

The issues with approximations of the simplicial complex primarily arise from feature loss. SimBa [23] reduces the initial complex to identify larger topological features within reasonable resources, but smaller features that do not directly contribute to identified topological features are not guaranteed to be stored and cannot be recovered. For general PH this is suitable, but smaller features for pattern detection and recognition may need additional detail. Alternatively, any sampling of data or clustering reduces the size of the input point cloud which will remove 0-dimensional features (connected components) and possibly higher dimensional features.

Recovery of lost features is not a priority of approximations to the PH, as the main goal has been to identify large features representing the homology groups of the point cloud. Recent studies have found use for smaller features in identifying small perturbations or differences between similar point clouds. In particular, Bendich *et al* [13] uses brain artery data to classify patients based on the small differences of the persistence intervals observed between scans of different patients.

Finally, the error induced from approximating the PH could completely mask the small differences between persistence intervals. In big data where the true persistence intervals can not be computed, the approximations may not give accurate results, in which case the analysis with TDA may not be viable. Convergence on the true persistence interval should be the goal of any approximation method to attempt to retrieve the most accurate barcodes with the resources available. The approach described in this paper builds a scalable approximation of the PH with analysis of convergence to the true persistence intervals present in big data.

IV. Data Reduction and Partitioning

This paper presents a method to apply both partitioning and data reduction for the computation of PH. In particular, the partitioning organizes a large point cloud of data into regional partitions for computing PH. The partitioning is coupled with a computation of PH on a data reduced representation of the original point cloud to provide an approximation of the PH of the original point cloud. The approach permits the computation of PH on much larger point clouds than previously possible. The remainder of this section will use the term "identify a topological feature" to indicate that the PH computation will

Original Camel triangular mesh point cloud

Reduced Camel triangular mesh point cloud

Fig. 1. Reduction of Camel triangular mesh point cloud with k-means++ cluster centroid replacement

return $< birth, death >$ intervals that represent a topological feature in the point cloud.

A high level outline of the steps in the method of this paper is described as follows:

1) Partition the point cloud P so that each point is assigned to one and only one partition. All points in the original point cloud must be placed in some partition. Let M be the maximum number of points in $\mathbb{R}^d$ for which the computation of PH can be performed. The targeted number of partitions for this step must be bounded by M. While some partitioning methods will results in faster and more complete solutions, as a general rule most d-dimensional spatial partitioning approaches will work. Partitioning is defined fully in Section IV-A.
2) Let P' be a set of points in $\mathbb{R}^d$ such that the elements of P' are composed of the geometric centers (centroid) from each partition. P' is the reduced data for approximating PH. This computation will identify the barcodes for the larger topological features in the point cloud. The smaller features will be identified by a regional PH computation described in step 3.
3) For each partition defined in step 1, compute PH on a region of points within and around the partition (these PH computations can occur in parallel among each other and concurrently to steps 2 and 4). The size of this region, to ensure identification of all topological features in dimension 2 and above, is slightly larger than the points in the contained partition. This additional size parameter is defined below. If any extended partition contains too many points for a PH computation, a recursive re-partitioning/data reduction step can be performed and the results returned as the PH result for the partition. A description of computing the regional PH in a partition is contained in Section IV-B.
4) For each topological feature identified in step 2 above, *upscale* the data in the partitions on the boundary of the feature and recompute the PH on the restored data from the partitions on the feature boundary (this step can be performed in parallel for each feature). Upscaling can be performed iteratively (by repartitioning/upscaling) to grow the restored data in cases where a one step upscaling presents too many points for computing PH. While the total number of points forming the convex hull[2] of the topological feature could, in rare cases, exceed the ability to compute PH on all of the boundary points; a subset of the points from the surrounding clusters can be used to partially upscale in order to reduce the error bounds of the computed homology. Further analysis of the upscaling of identified features is contained in Section IV-C.
5) This method may produce duplicate persistence intervals for the topological features in the original point cloud. Any duplicate features found by the regional PH computations can be eliminated by creating an arbitrary total order on the points in the original data set and having each regional computation report only $< birth, death >$ intervals for topological features where the lowest ordered point in the convex hull of that feature is a member of a partition of that regional PH computation. A final step is to remove duplicate $< birth, death >$ intervals returned from the centroid based PH computation that are also discovered in a regional PH computation. The regionally computed $< birth, death >$ interval is preserved as it will generally have a more precise computation of the feature $< birth, death >$ interval than the centroid based PH computation. The persistence interval merging of duplicate and overlapping features can be found in Section IV-E.

A. Partitioning

The first step in this approach is to partition the points in the original point cloud. While some partitioning methods may result in faster and more complete solutions, as a general rule

[2]While the convex hull of a set P is generally defined as the boundary around the smallest convex set of points that contain P, this paper uses the term convex hull around a topological feature (a hole, void, or loop) to denote the set of points forming the boundary around the topological feature (in all dimensions of that void) precisely at the minimum ϵ distance when the feature first appears.

most d-dimensional spatial partitioning approaches will work. The partitioning should define no more than M partitions and must place each point into a unique partition. More precisely, let $\hat{P} = \{p \mid p \subset P\}$ be a partitioning of P, then $\forall p, q \in \hat{P} \mid p \neq q, p \cap q = \emptyset$ and $\cup_{p \in \hat{P}}\ p = P$.

From this partitioning, the algorithm will then select a single representative point from each partition to define a new (reduced) point cloud. While any data point (actual or representative) from each partition can serve this purpose, this work will examine the specific use of the geometric center of each partition, where the geometric center is the mean of each dimension of all points in that partition. More precisely, let P' be the set of geometric centers of the partitions $\hat{P}$, then if $p_i \in P'$ is the centroid for partition $\hat{p}_i \in \hat{P}$ then $p_i = \frac{\sum_{q \in \hat{p}_i} q}{||p_i||}$. In the remainder of this paper, the term *centroid* will be used to denote the geometric center of a partition.

The objective in this step is to discover a point cloud P' such that the large topological features of P are also in P'. This desired similarity is necessary only for the larger topological features (where larger is formally defined below); the smaller topological features will be discovered and characterized by their birth/death times during the regional PH computations described in Section IV-B. In the remainder of this paper, the following terms will be used without further definition:

- $\hat{P}$, the partitions,
- P', the centroids,
- r_i, the distance from the partition centroid, $P'_i \in P'$, to the most distant point in that partition, and
- $r_{max} = max(r_i)$, the maximum r_i of all the partitions.

The use of the smaller point cloud P' to estimate the PH of P will identify the larger topological features. In particular let B be the boundary of points in the complex circumscribing a $d \geq 2$-dimensional topological feature and let $s_B = max(distance(b_i, b_j)) \forall b_i, b_j \in B$, then we define the term "large topological feature" to be any feature with $s_B > 2r_{max}$. Informally, a large topological feature has a diameter that cannot be contained within the largest partition defined in $\hat{P}$. Any topological feature with a diameter smaller than $2r_{max}$ may or may not be identified during this step. In particular, the identification of the smaller topological feature by the estimation of PH will depend specifically on whether that feature lies between the boundaries of the partitions or if it is wholly contained within a partition. Fortunately, as described in Section IV-B, the regional computations of PH on each partition will identify all smaller topological features.

Ideally the partitioning and data reduction steps described above will cooperatively result in a centroid-based data set P' that is, for the large topological features, topologically similar to that of P. However, this is not required. Furthermore, while traditional uses of PH focus on large topological features, in cases where the large features are insignificant [12], [13], P' and its PH computation can be skipped. The regional PH computations will compute the PH of the small features.

In the discussion above, partitioning and data reduction are presented as a coupled process to extract smaller collections of data vectors to be used in the computation of PH. However, it is not necessarily that the geometric centers of the partitions also form the points for the reduced data. This is done for convenience of the exposition and to streamline the formal development. Technically they can be performed independently and there is nothing the prevents some form of partitioning the data and independently using something such as random selection [7] for the data reduction step.

B. Smaller Topological Feature Identification

Since the point cloud P' will identify the $2 < k < d$ dimensional topological features containing a diameter $d_i > 2r_{max}$, more work may be needed to identify topological features with smaller diameters. This can be achieved by computing PH at each partition. However since a feature can originate near the boundary of a partition and extend outside of the partition, it becomes necessary to include points beyond the partition boundary so that the smaller features are all properly identified. Thus, in order to properly identify these smaller features, the PH computation must be performed with all points from P that lie within a distance of $r_i + 2r_{max} + \epsilon_{max}$ from the partition centroid $P'_i \in P'$. That is:

Theorem 1. *All $2 < k < d$ dimensional topological features containing a convex hull of diameter $d_i < 2r_{max}$ in P can be identified by performing a PH computation corresponding to each partition $p_i \in \hat{P}$ such that the points included in the PH include all the points in the partition plus any point in P that lie within the hyper-sphere of radius $r_i + 2r_{max} + \epsilon_{max}$ centered at the centroid for that partition.*

The regional PH computations can all be performed in parallel. Of course, if the total number of points contained in the bounding radius defining the regional PH computation exceeds M, then it may be necessary to recursively partition the points of the regional space and iterate this algorithm to extract the persistence intervals for that region.

C. Upscaling

Partitioning, as described in IV-A, generates a representative point cloud P' of an original point cloud P that is topologically similar regarding large features. For each feature identified through the PH of P', there exists a mapping back to P that indicates the source points represented in the approximated persistence. This reveals that any feature approximated in P' is also a feature in P that can be mapped to constituent points forming the true feature. By determining the topological features in P', the constituent points can be further analyzed to refine the $< birth, death >$ interval in a subset of P, removing data not contributing to the approximated feature for faster computation. This process is called *upscaling*.

PH emits boundary information not typically examined during the computation of persistence intervals. More precisely, when loops in the data are computed in the boundary matrix reduction, the boundary points forming a loop are also available if tracked through the reduction step. The boundary points represent the critical path forming a loop for the corresponding

persistence interval. Examining the boundary of a persistence interval can be used to extract that individual boundary's points for further identification of the feature.

In some cases a partitioning, P', may shift the dimension of an identified topological feature (into either a higher or lower dimension). Certain representative points of classifications may end up coplanar in one dimension, causing reduction of the original dimension of the topological feature. In other cases, the representatives may create features in a higher dimensions than in P, and cannot be rectified without utilizing the mapping from P' back to P.

Upscaling of significant topological features identified in P' aids in recovering features classified in incorrect dimensions. Regardless of the dimension an approximated feature of P' is initially identified in, the upscaling step recomputes the PH over P, utilizing the constituent boundary points of the feature. This step ignores the previous persistence interval calculated with the approximation and will use points from P to produce an accurate persistence interval, or multiple persistence intervals describing the boundary. This also indicates when features shift a dimension, either higher or lower, the upscaling step will recompute the PH around that feature regardless of the P' classifications of the approximated feature.

The upscaling of significant topological features can provide significant improvements to the persistence intervals generated by a point cloud regardless of the initial size of the data-set. With a well-chosen partitioning algorithm the significant topological features can be preserved and upscaled to produce more accurate persistence intervals beyond the limits of current tools for computing PH.

D. Upscaling Limits

There are limitations to the effectiveness of upscaling in the context of the geometric structure of points within the point cloud. More precisely, let B be the boundary of centroid points around a topological feature and let C be the set of clusters from which all $b_i \in B$ are originate. Then the upscaling step is performed on all points in all members of C. Further let q_{max} be the maximum radius for all the clusters in C (where the radius is defines as the distance from the centroid any point in C). Then,

Theorem 2. *The upscaling step has a worst case error approximation of the approximate $< birth, death >$ interval of a large topological feature to $2q_{max}$.*

Upscaling has the potential to reduce the error bounds for the large topological features to zero. However, there are geometric situations where this reduction is not complete. In particular, consider a feature shaped as two intersecting n-spheres (S_0 and S_1) with an opening between them of distance d. Furthermore assume that there is a circumscribing path around S_0 and S_1 such that the path can be connected at distance $d' < d/2$. Assume that the partitioning is such that the opening between S_0 and S_1 is covered by two identical spherical partitions (P_0 and P_1) of radius $d/2$ each centered at the two closest boundary point where S_0 and S_1 intersect and meeting at the center of the opening between S_0 and S_1. Then the upscaling step will stop with a $< birth, death >$ interval for S_0 and S_1 that is bounded by a birth distance $\geq d/2$ even though there should be a single persistence interval for a feature including both S_0 and S_1 with a $birth$ value at d'.

E. Merging Duplicate Topological Features

It is possible that a the PH computation on P' and the regional partitions identify the same feature. Furthermore, since the centroid based PH is approximate, it may not be possible to filter by an exact match of the boundary to the regional results. Thus, an *approximately equal to* relation will be used to remove duplicates.

In addition, it is possible that two different regional PH computations may locate the same small feature (if the feature is contained within an intersecting region between two more more regional PH computations). Fortunately, any duplication of features found by different regional PH computations can be filtered out by defining a total order on all points in P. With this order, it is possible to restrict the reporting of a topological feature by a regional computation only when the lowest ordered point on the boundary of the identified feature lies within the original partition for which the regional PH computation is performed. This will remove these duplicates.

V. Implementation Strategy and Techniques

The implementation of the described method can become complex in order to correctly identify and map persistence intervals between several separate pipelines, specifically when dealing with regional PH. To account for this complexity, a preliminary technique was evaluated to focus on: (i) upscaling of connected components, (ii) upscaling of independent boundary features, and (iii) upscaling of individual centroids. Each function is independent of the others and provides opportunities for parallelism to increase performance beyond the experimental results in this study.

The simplest function is to compute the connected components of the original data-set, P. With a low d_{max} and ϵ the persistence intervals can be output up to M points, where M is primarily dependent on available memory resources. In most cases, the original data can be computed at $d_{max} = 0$ for millions of points. This processing on P will give the true 1-dimensional topological features (connected components) within the data-set.

Computing the PH on the individual partitions will identify additional topological features within a centroid (where $birth < death < r_{max}$) that were missed during the first pass. For examining the additional features in a partition, only features of dimension 2 or higher are considered, as the connected components are computed individually.

The constituent boundary points provide a mapping of large features identified in the first pass back to their initial centroids. Each centroid is the representative point of a respective partition of the original point cloud, and can be upscaled to provide more accurate persistence intervals of large features.

Each of these methods provides additional features that can be merged into a single set of persistence intervals: the true 1-dimensional features cover connected components of the complex; the upscaled centroids will find smaller, higher dimensional features embedded in the reduction; the upscaling of the large feature boundary points will refine persistence intervals found in the approximated PH. During upscaling three sets of persistence intervals are collected: the connected components, the upscaled features, and the upscaled centroids. The connected components represent the complete set of 0-dimensional features identified in the point cloud. The upscaled features contain subsections of the point cloud with identified features through approximation, and only are concerned with higher dimensional (> 0) features. The upscaled centroids handles the remainder of the centroids not included in the other boundaries to compute local features smaller than r_{max}. All of these data-sets are independent and merging does not require any special mapping or comparison of the data.

VI. Experimental Results

This section evaluates the described approach to data reduction through clustering alongside upscaling to refine the persistence intervals. The experiments performed analyze both partitioning and upscaling as a means to minimize the induced error in previous sampling experiments performed by Moitra *et al* [8]. Each test requires multiple steps: partitioning the data to generate clusters and indexes for the mappings, computing PH on the reduced point cloud, and upscaling the approximated results to refine error in $< birth, death >$ intervals. The results are analyzed to characterize the effects of clustering on the PH and the degree of rectification upscaling yields in producing more accurate $< birth, death >$ intervals.

In all cases k-means++ was used to cluster the original data-set. Previous studies show that partitioning with k-means++ preserves the relative shape of data regardless of varying density throughout the point cloud [8]. For the upscaling approach described the identification of large features in the initial processing of PH is vital to the accuracy of the output persistence intervals. Enough centroids must be used when initially processing the data to properly identify large features. Centroid counts are used throughout the experimental results to describe the size of the reduced data set in the context of upscaling. The processing also included output of centroids for each identified cluster along with index labels to map the original points to their constituent centroids. Once data was preprocessed and classified with k-means++, the centroids were used for PH computation.

PH computation requirements for upscaling include output of the boundary points forming significant features in the point cloud. Eirene was used for this purpose, as other PH tools do not track and resolve the original points contributing to topological features. Eirene outputs the boundary points forming significant features as indexed sets referring to each original point passed to the library; in our case, the index of the centroid is identified for each boundary. The boundary centroids were recorded to file for upscaling along with the persistence intervals for comparison and analysis.

Upscaling is executed as a post-processor, requiring the original data, the centroids and labels, and the boundary centroids contributing to significant features. The upscaling function can use any PH library, as boundary extraction from the PH boundary matrix is not immediately necessary. For consistency in application requirements Eirene was used for PH computation steps in the approach.

All experiments were performed on an Intel(R) Xeon(TM) E5-2670 CPU @ 2.60GHz with 64GB of RAM and an additional 64GB of swap. The Scikit-Learn implementation of k-means++ was used to cluster the data, outputting centroids and labels. The Eirene library, executed in Julia 1.1.0, was used for computing the PH of the reduced data and subsequently used for upscaling. Both GUDHI and Ripser were included to collect persistence intervals at maximum vector sizes for comparison. All results were organized and stored for post-processing analysis of persistence intervals, cluster characteristics, and upscaling accuracy.

Analysis of the experiments involved computing the continuous heat-based kernel metric [25] to compare output persistence intervals from different partitions of the point cloud. For comparing results between reduced and upscaled data the continuous heat-based kernel provides a multi-scale kernel designed for topological machine learning, giving a stable metric for evaluating persistence diagrams in comparison. The continuous heat-based kernel in this experiment is used to give a general degree of difference between compared persistence intervals.

The upscaling step was first evaluated to demonstrate the effectiveness of data reduction through partitioning coupled with upscaling. Point clouds were chosen where the source data could be computed with existing PH tools. Evaluation of the source data provides a baseline, or ground truth, to compare all partitioned and upscaled data against. Persistence intervals that match the ground truth will have continuous heat metric of 0. The implemented approach is expected to have some error due to features that are not wholly contained within a partition, but should have a significant impact to the continuous heat metric when compared to the persistence intervals of the reduced data.

To explore the effectiveness of upscaling the Stanford Dragon and TwoCircles point clouds were used. The Stanford Dragon data-set has $2,000$ vectors in R^3 around the surface of a 3-dimensional dragon model. Study of the Dragon point cloud has been found in several topology and machine learning applications, specifically around object recognition and differentiation [6]. The TwoCircles data-set is a $2,000$ vector point cloud of two independent circles in R^2. The TwoCircles data-set provides a synthetic data-set for understanding the impact of low dimensional features (loops) on the upscaling approach. PH can be computed at full scale for both of these point clouds, enabling comparison to the ground truth persistence intervals.

Performance characteristics for the upscaling technique are shown in Table I. Individual times for partitioning, approx-

Dataset Name	P Persistence Time (s)	Partitions	Partitioning Time (s)	P' Persistence Time (s)	Upscale Time (s)	Total Time (s)
Dragon	208.19	500	1.72	0.40	155.25	157.37
twoCircles	122.56	500	1.25	0.62	61.06	62.93
Lion (Max 1500)	508.20	500	4.10	3.52	119.84	127.44
Camel (Max 1500)	178.68	500	24.44	1.68	80.49	106.61
Circles (Max 150)	1740.78	100	0.61	5.00	77.75	83.36

TABLE I
PERFORMANCE OF THE EXPERIMENT FOR EACH EVALUATED DATA-SET.

	Reduced Size	Reduced Heat Kernel Distance	Reduced Betti Count	Upscaled Heat Kernel Distance	Upscaled Betti Count
Stanford Dragon	100	8271.51	150	1680.70	2301
	250	7942.82	389	909.56	2146
	500	6804.43	783	6688.97	2047
	750	5631.02	1146	2762.19	2020
	1000	4596.11	1509	2791.65	2007
	1250	3566.31	1833	2814.04	2004
	1500	2814.87	2145	2820.00	2001
	2000	0	2701	NA	NA
TwoCircles	100	3575.62	110	DNF	DNF
	250	3192.58	272	3082.47	2257
	500	2587.92	540	2015.35	2021
	750	1985.96	804	DNF	DNF

TABLE II
DIFFERENCES IN THE PERSISTENT DIAGRAMS AFTER REDUCTION AND UPSCALING, AS COMPARED TO THE KNOWN PERSISTENCE INTERVALS FOR THE FULL SCALE DATA-SET. (DNF: DID NOT FINISH)

imated persistence on the reduced data, and upscaling are shown separately. The performance increase from partitioning and upscaling provides a considerable argument for processing the PH of big data. In the cases of the data sets explored later (Circles, Lion, and Camel), the maximum point cloud is limited by experiments that did not finish with current PH tools on the available systems. The time for comparison in these cases is the time of the maximum computable PH.

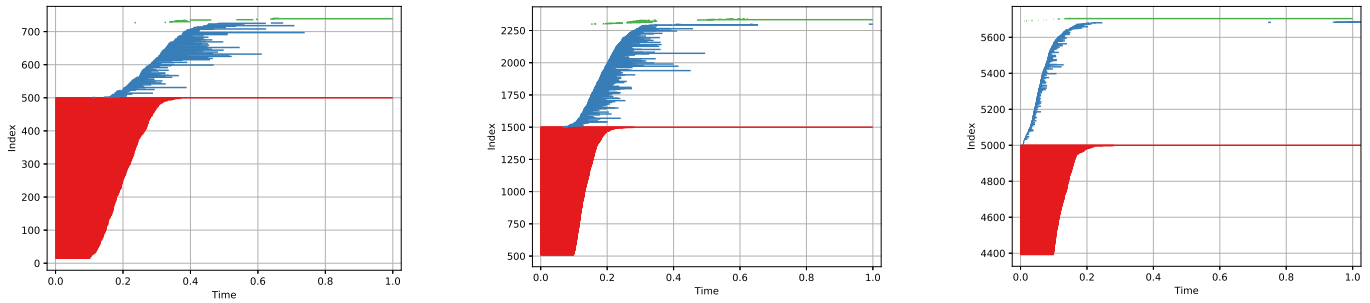
Performance gives an initial indication that data reduction and partitioning combined with upscaling brings faster topological feature identification, and clearly reduces the resource requirements for computing topology on large data sets. The described technique also needs to correctly identify features in point clouds and improve on the approximated persistence. Optimal results for upscaling the approximated topological features requires the partitioning of data to preserve the significant topological features in the original data-set, P. Several different partitions were examined for each data-set to observe optimal upscaling attempts at different reduction levels.

Experimental results for the Stanford Dragon and Circles data-set are shown in Table II. Several interesting results can be found for both data-sets. In the Stanford Dragon data-set, lower reductions such as 100 and 250 points with k-means++ provided results better than all others in terms of the continuous heat metric. This indicates that a low number of clusters preserves the topology of the source data well, and can be upscaled with marginal error induced to the persistence diagram compared to reduction alone. An interesting case is also found in the TwoCircles data-set.

The results from the first experiment show that the implemented upscaling method does reduce the induced error from partitioning when the partitioning preserves the underlying topological structure of the point cloud. There are still limitations to the approach, and some minor error will still exist, but upscaling does refine the topological features found from an approximated PH in the partitioned space, P'. To examine the limitations of this refinement several other data-sets beyond current PH tools were evaluated.

While the approach still remains approximate, the refinement of $< birth, death >$ intervals and identification of small topological features provides additional insight not currently available when examining PH. Circles is a generated data-set with 2-dimensional circles embedded in R^{10}. There are 1500 vectors in the Circles data-set, permitting current TDA tools to compute the ground truth $< birth, death >$ intervals. The dimensionality of the Circles data-set is one of the primary reasons it was chosen, as current tools do not scale well in higher dimensions. The Lion and Camel data-sets are both from the UCI Machine Learning repository and describe the boundaries of their respective models in $\mathbb{R}^3$ [26]. The Lion model has $4,999$ vectors, making it just on the boundary of being evaluated with current tools. The Camel data-set has $26K$ points, requiring significantly more memory to build the complex and extract the persistence intervals.

Ground truths can not be utilized with the Circles, Lion, and Camel data-sets due to the number of simplices generated from the original point cloud is beyond limitations of current persistent homology tools. For this reason, the upscaled persistence intervals can not be quantitatively compared to the actual persistence intervals. Comparing the upscaled persistence intervals to the maximum computable reduction for each of these data sets provides uncertain results, as the upscaled persistence represents the entire data set while the reduced data-set is only



500 Vector Barcodes | 1500 Vectors (Max PH) Barcodes | Upscaled Barcodes

Fig. 2. Refinement of the Lion persistence intervals shown through an original partitioning, a maximum partitioning based on PH limitations, and an upscaled set of intervals for the full data-set. The green lines represent B_2 barcodes, the blue B_1, and the red B_0; B_0 persistence intervals shorter than 0.1 were filtered to provide a clearer picture of the higher dimensional topological features.

looking at a partitioned space, P'.

Because a metric can not be used in these cases to indicate refinement of data reduction through partitioning and upscaling, persistence images are used to give a general idea of each of the homology groups identified in the point cloud. Figure 3 shows the persistence images for the Lion data-set at original reduction level, maximum point reduction level, and after upscaling. Each subset is divided into the H_0, H_1, and H_2 to show the impact of upscaling on each topological group.

The Lion persistence images were chosen because they display the refinement of the topological features identified in P', of 500 vectors, that align more closely with the maximum point reduction level, 1500 vectors. A similar conclusion can be drawn from the barcodes for the same Lion results in Figure 2. At all 3 dimensions, the features present in the persistence intervals of the reduced data are significantly different than the maximum point reduction and the upscaled features. In the persistence images of Figure 3, the reduced persistence in H_0 is high, reduced persistence in H_1 is high and to the right, and reduced in persistence H_2 is slightly to the right. The similarity between the upscaled and maximum point reduction data is much closer, indicating the features present in both point clouds are roughly equivalent. Analysis with the full point cloud would have more dense regions in H_0 and H_1 due to the small features removed during the reduction step, thus the radius of upscaled persistence diagrams is much smaller.

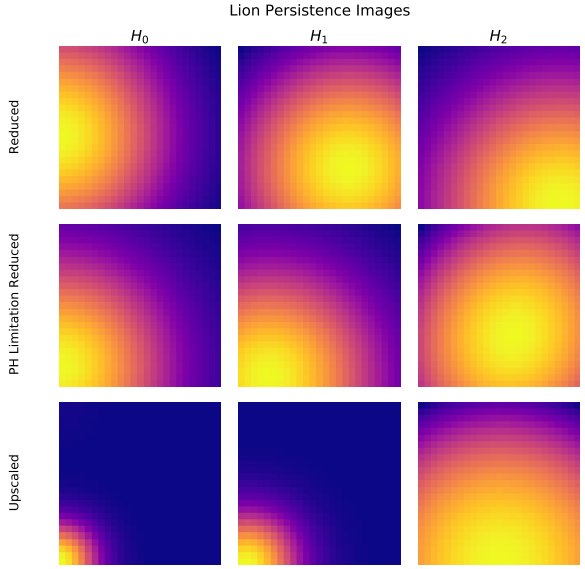


Fig. 3. Persistence images for the Lion data-set. Columns show the homology groups for each evaluated set of persistence intervals. For rows, Reduced is the original reduction of the data; PH Limitation Reduced is the max vector size PH can be computed over with current tools; Upscaled is the generated persistence intervals after upscaling the reduced data.

VII. Shortcomings of Upscaling

The upscaling approach described in this paper is straightforward and requires little effort to understand and implement. Less obvious nuances, such as loss of topological features due to partitioning, are not handled through this experimental simplistic approach. In general there are several factors that can significantly affect the topological features identified during the initial approximation phase that determine the effectiveness of upscaling. The partitioning algorithm used for data reduction needs to preserve the topological features present in the data for upscaling to be viable. The number of points classified to a single cluster can also shift the results of upscaling, or render the upscaling attempt useless due to the number of points classified together. Finally, changes to features are not immediately addressed for all cases described in Sections IV-B. This section describes in detail the known shortcomings and suggestions for improvement of upscaling through several optimizations.

Selection of a correct initial partitioning algorithm to preserve significant topological features is not inherently obvious, though can be adjusted for differing point clouds manually. While the experiments described in this paper focus on the k-means++ algorithm to generate centroids and associate the original point cloud to the partitioned point cloud, other partitioners may provide better results in certain situations. Even with a well-chosen partitioning algorithm, the significant topological features in the source point cloud can be lost if too few partitions are identified.

To account for the consequential effect initial data reduction has on upscaling the approximated persistence intervals a multi-partitioning approach that samples several data reduction

levels and only upscales significant features (*i.e.,* $d > 0$) may identify additional features lost when only examining the point cloud at a single reduction level. With multiple classifications of the point cloud contributing to the realized features for upscaling, both large and small features could be identified and upscaled simultaneously, providing a more refined result.

Upscaling does provide computation of PH beyond the current limits of TDA libraries, but is still limited by system resources in all cases. If an extremely large data set is used, say $100,000$ vectors, and classified into 100 centroids each centroid will contain somewhere around $1,000$ points. Persistent homology of the classified points may not be accessible, in which case the persistence interval is thrown out. To account for larger data sets and unbalanced partitions of the original point cloud, an iterative approach may be used to compute PH up to the system resource limitations. If the centroid of $1,000$ points can not be computed, the data would be reduced again and evaluated as another upscaling step.

Iterative upscaling could provide a mechanism for automatically scaling to the system resources based on the complexity of the point cloud. Examining the persistence intervals at different vector sizes and epsilon values could identify additional features not immediately recognized through partitioning. Iterations could also track boundaries and constituent points beyond their internal upscaling process, joining multiple parallel efforts to refine the persistence intervals that merge into a single, more complete output. The design of iterative upscaling would utilize many of the observations detailed in upscaling and address the issues of feature loss and parameter selection when analyzing unknown data-sets.

VIII. Conclusions

Data reduction and data partitioning have many benefits in analysis of big data that extend beyond measure. These topics have been studied extensively in TDA to provide more efficient ways to identify topological features within a point cloud, often using PH. Accelerating the performance and reducing the resource requirements for computing PH is necessary to enable automation and extension of topology into machine learning and the future of data analysis. The method presented in this paper extends previous study of partitioning for PH by introducing an upscaling process that approximates and refines persistence intervals in an efficient and robust manner. Experimental results with the primary features of upscaling show the process can greatly improve the identified persistence intervals in known point clouds, and can extend beyond current tools for approximating persistence intervals of big data.

References

[1] R. Ghrist, *Elementary Applied Topology*. Createspace, 2014.

[2] F. Chazal and B. Michel, "An introduction to topological data analysis: fundamental and practical aspects for data scientists," *ArXiv e-prints*, Oct. 2017.

[3] R. Ghrist, "Barcodes: The persistent topology of data," *Bulletin of the American Mathematical Society*, vol. 45, no. 1, pp. 61–75, 2008.

[4] P. Y. Lum, G. Singh, A. Lehman, T. Ishkanov, M. Vejdemo-Johansson, M. Alagappan, J. Carlsson, and G. Carlsson, "Extracting insights from the shape of complex data using topology," *Scientific Reports*, vol. 3, Feb. 2013.

[5] G. Singh, F. Memoli, and G. Carlsson, "Topological methods for the analysis of high dimensional data sets and 3D object recognition," in *Eurographics Symposium on Point-Based Graphics*, M. Botsch, R. Pajarola, B. Chen, and M. Zwicker, Eds. The Eurographics Association, 2007.

[6] N. Otter, M. A. Porter, U. Tillmann, P. Grindrod, and H. A. Harrington, "A roadmap for the computation of persistent homology," *EPJ Data Science*, vol. 6, no. 1, Aug. 2017.

[7] F. Chazal, B. T. Fasy, F. Lecci, B. Michel, A. Rinaldo, and L. Wasserman, "Subsampling methods for persistent homology," in *International Conference on Machine Learning*, ser. ICML 2015, Lille, France, Jul. 2015. [Online]. Available: https://hal.archives-ouvertes.fr/hal-01073073

[8] A. Moitra, N. Malott, and P. A. Wilsey, "Cluster-based data reduction for persistent homology," in *IEEE International Conference on Big Data*, Dec. 2018, pp. 327–334.

[9] D. R. Sheehy, "The persistent homology of distance functions under random projection," in *Proceedings of the Thirtieth Annual Symposium on Computational Geometry*, ser. SOCG'14. New York, NY, USA: ACM, 2014, pp. 328–334.

[10] V. de Silva and G. Carlsson, "Topological estimation using witness complexes," in *Eurographics Symposium on Point-Based Graphics*, ser. SPBG '04, M. Gross, H. Pfister, M. Alexa, and S. Rusinkiewicz, Eds. The Eurographics Association, 2004.

[11] K. N. Ramamurthy, K. R. Varshney, and J. J. Thiagarajan, "Computing persistent homology under random projection," in *2014 IEEE Workshop on Statistical Signal Processing (SSP)*, Jun. 2014, pp. 105–108.

[12] H. Adams, T. Emerson, M. Kirby, R. Neville, C. Peterson, P. Shipman, S. Chepushtanova, E. Hanson, F. Motta, and L. Ziegelmeier, "Persistence images: A stable vector representation of persistent homology," *Journal of Machine Learning Research*, vol. 18, no. 1, pp. 218–252, Jan. 2017. [Online]. Available: http://dl.acm.org/citation.cfm?id=3122009.3122017

[13] P. Bendich, J. S. Marron, E. Miller, A. Pieloch, and S. Skwerer, "Persistent homology analysis of brain artery trees," *The Annals of Applied Statistics*, vol. 10, no. 1, pp. 198–218, Mar. 2016.

[14] G. Carlsson, "Topology and data," *Bulletin of the American Mathematical Society*, vol. 46, no. 3, pp. 255–308, Apr. 2009.

[15] H. Edelsbrunner and J. Harer, *Computational Topology, An Introduction*. American Mathematical Society, 2010.

[16] U. Bauer. (2019) Ripser. The Technical University of Munich. [Online]. Available: http://www.cs.umd.edu/ mount/ANN/

[17] C. Maria, J.-D. Boissonnat, M. Glisse, and M. Yvinec, "The gudhi library: Simplicial complexes and persistent homology," INRA, Tech. Rep. RR-8548, 2014. [Online]. Available: https://hal.inria.fr/hal-01005601v2

[18] G. Henselman. (2019) Eirene: julia library for homological persistence. [Online]. Available: https://github.com/Eetion/Eirene.jl

[19] J.-D. Boissonnat and C. Maria, "The simplex tree: An efficient data structure for general simplicial complexes," *Algorithmica*, vol. 70, no. 3, pp. 406–427, Nov. 2014.

[20] C. Chen and M. Kerber, "Persistent homology computation with a twist," in *Proceedings 27th European Workshop on Computational Geometry (EuroCG'11)*, 2011, pp. 197–200.

[21] U. Bauer, M. Kerber, and J. Reininghaus, "Clear and compress: Computing persistent homology in chunks," in *Topological Methods in Data Analysis and Visualization III*. Springer International Publishing, 2014, pp. 103–117.

[22] M. Mrozek and B. Batko, "Coreduction homology algorithm," *Discrete & Computational Geometry*, vol. 41, no. 1, pp. 96–118, Jan. 2009.

[23] T. K. Dey, D. Shi, and Y. Wang, "Simba: An efficient tool for approximating rips-filtration persistence via simplicial batch-collapse," *24th Annual European Symposium on Algorithms (ESA 2016)*, 2016.

[24] J. A. Barmak and E. G. Minian, "Strong homotopy types, nerves and collapses," *Discrete & Computational Geometry*, vol. 47, no. 2, pp. 301–328, Mar. 2012.

[25] J. Reininghaus, S. Huber, U. Bauer, and R. Kwitt, "A stable multi-scale kernel for topological machine learning," in *2015 IEEE Conference on Computer Vision and Pattern Recognition*, ser. CVPR, Jun. 2015, pp. 4741–4748.

[26] R. W. Sumner and J. Popovic, "Mesh data from deformation transfer for triangle meshes," 2004. [Online]. Available: https://people.csail.mit.edu/sumner/research/deftransfer/data.html