MOG: Mapper on Graphs for Relationship Preserving Clustering

Mustafa Hajij, Bei Wang, Paul Rosen

(a) The mapper graph summary (top) along with the original graph (bottom)
(b) Edges in the mapper graph represent relationships between clusters

Fig. 1: (a) The mapper graph (top) provides a mechanism for extracting a compact and meaningful summary that captures the “shape” and the main underlying structure of the original graph (bottom). Each node of the mapper graph represents a community or a cluster in the original graph. (b) In addition to capturing clusters in the original graph, mapper also provides the relationships among these clusters.

Abstract—The interconnected nature of graphs often results in difficult to interpret clutter. Typically techniques focus on either decluttering by clustering nodes with similar properties or grouping edges with similar relationships. We propose using mapper, a powerful topological data analysis tool, to summarize the structure of a graph in a way that both clusters data with similar properties and preserves relationships. Typically, mapper operates on a given data by utilizing a scalar function defined on every point in the data and a cover for scalar function codomain. The output of mapper is a graph that summarize the shape of the space. In this paper, we outline how to use this mapper construction on an input graphs, outline three filter functions that capture important structures of the input graph, and provide an interface for interactively modifying the cover. To validate our approach, we conduct several case studies on synthetic and real world data sets and demonstrate how our method can give meaningful summaries for graphs with various complexities.

Index Terms—Topological data analysis, mapper, graph clustering

1 Introduction

Graphs are a common data type, yet visualizing them remains a challenging problem. When considering node-link diagrams, the interconnectedness of the nodes causes edges to cross, which leads to visual clutter that makes understanding the structure difficult. Techniques such as edge bundling, motifs, and clustering have been proposed to reduce the visual clutter. In the case of edge bundling, decluttering occurs by grouping edges with similar relationships. For techniques such as motifs and clustering, clutter is reduced by grouping nodes with similar properties.

In this work, we present a new approach to graph decluttering that clusters nodes based upon certain properties, while maintaining a strong notion of relationship. Our method relies on a topology-inspired construction called mapper [80]. Mapper is a tool from Topological Data Analysis (TDA) that provides a topological summary of the data [14].

Thanks to its intuitive construction and its applicability on a wide variety of data-related problems, the construction of mapper has become one of the most successful tools in TDA. Mapper has been applied in many areas including visualization of high dimensional data [58], pattern recognition of point clouds [15], tracking resilience to infections [82], and many others [59, 66].

The mapper construction operates as an approximation tool of a general topological space. The construction operates by mapping the topological space via a “lens”, or a filter function, to another domain called the parametrization space. The properties of the filter and the parameterization space are then utilized to obtain an approximation of the original space that both clusters and preserves the relationship between clusters.

Our approach works as follows: Starting with an undirected, weighted, or unweighted graph, a filter function is calculated per node, for example average geodesic distance, density, eigenfunctions, etc. A cover, which describes the cluster construction of mapper is selected.

Finally, the mapper graph is constructed and visualized.

The mapper graph, as seen in Figure 1(a) demonstrates 2 important properties. First, the nodes of the mapper graph represent clusters from the graph. Second, the edges of the mapper graph represent relationships between clusters, specifically which clusters overlap in the original graph. Furthermore, we provide an interactive mechanism for modifying the cover, in order to better explore the underlying graph structure. These features combined provide the capability to more
easily understand the structure of a graph under the “lens” of a filter function.

In summary, the contributions of this paper are:

- Generating a property and relationship preserving summary of graphs by applying mapper to the graph;
- Enabling exploration of the structure by providing an interactive cover and graph modification mechanism;
- Demonstrating the capabilities of mapper on graphs using 3 different filter functions, namely, average geodesic distance, density, and eigenfunctions of the graph Laplacian; and
- Case studies on synthetic and real graph data showing the effectiveness of our method at finding meaningful summaries in graphs.

2 Prior Work

We review prior work in graph visualization, node-link diagrams specifically, graph clustering, and the application of Topological Data Analysis to graphs.

2.1 Graph Visualization

For a comprehensive overview of graph visualization techniques, see [35]. We provide a brief outline of the most relevant methods for visualizing graphs using drawing node-link diagrams, which are utilized in many popular graph visualization software applications, such as Gephi [8], Graphviz [31], and NodeXL [41].

The problem of visual clutter in graphs has been extensively studied in the literature of graph visualization [30]. It has been addressed in 3 main ways, improved node layouts, edges bundling, and alternative visual representations.

The earliest graph layout method for node-link diagrams goes back to Tutte [83]. This was followed later by methods driven by linear programming [37], force-directed embeddings [34,44], embeddings of the graph metric [56], and connectivity structures [11,47,49,50]. Later methods created hybrid layouts driven by graph topology [2].

To reduce visual clutter on dense graphs, edge bundling can be used by rerouting edges with a common fate to overlap [42]. For massive graphs, hierarchical edge bundling has been utilized [35] and it scales to millions of edges, while divided edge bundling [76] tends to produce higher-quality results.

Other methods such as replacing nodes with an alternative visual representation has been used in the literature of visual cluster. These methods range from variations on node-link diagrams, such as replacing nodes with modules [29] or motifs [27], to more abstract representations, such as variants of matrix diagrams [25] or the abstract displays of the graph statistics [46].

2.2 Graph Clustering

Broadly speaking, our approach is closely related to graph clustering methods. The objective in graph clustering is to group the nodes of the graph together by taking into consideration the edge structure [74]. It is important to note that this is a different problem from clustering a set of graphs, where the structure similarity between a set of graphs is studied. Graph clustering algorithms are studied extensively. We only give a brief overview of the literature. See [17,50] for more details. The techniques that have been used for graph clustering are very diverse. These techniques include spectral graph clustering-based methods [21,35,51,87], similarity-measure based methods [81], global graph clustering methods [61,62], random walks models [45,71], and hierarchical graph clustering algorithms [10,73]. Some graph clustering literature is also aimed at directed graphs [12,84]. Clustering of graph edges is studied in [20,32]. Applications of graph clustering include community detection [38,63,64], analyzing the clusters of the global air transportation network [69], and scientific citation and collaboration [72].

2.3 Topological Data Analysis of Graphs

Over the last decade many concepts and tools from Topological Data Analysis have been introduced to the visualization community.

Persistent homology is the most notable tool of TDA that has been used to study graphs [26,29,43,68,69]. It has been applied to study graphs in numerous applications, such as collaboration [3,17] and brain networks [18,21,53,56,70].

Mapper [80] has been widely utilized in TDA for a number of applications [15,38,59,60,82]. Recently Mapper has witnessed major theoretical development that further adjudicate its use in data analysis [16,22].

Besides being theoretically justified, mapper generalizes other topological summaries such as the Reeb graph [60], the contour tree [80], split, and joint trees. All these construction have found enormous applications in data visualization and data understanding. Mapper is also the main software developed by Ayasdi, a company that utilizes methods inspired by topological construction in applications of data science.

To the best of our knowledge Mapper has not been utilized yet in the graph visualization.

3 Overview on Mapper Graph Construction

In this section, we apply the mapper algorithm to a graph and construct a multi-scale abstraction, referred to as a mapper graph, for summarization and exploration. We first introduce the mapper algorithm in its generality for a real-valued function with necessary but minimal topological notions. We then give an overview of the mapper construction pipeline for graphs.

3.1 A General Mapper Construction

Given a compact topological space $X$ that is equipped with a real-valued function $f : X \rightarrow \mathbb{R}$, the mapper construction (or mapper for short) provides a general framework to study $X$ which is parametrized with respect to $f$. The function $f$, commonly referred to as a filter function, plays the role of the lens, through which we look at the properties of the space, and different lenses provide different insights [7]. This is one of the key ideas behind mapper.

Nerve. An open cover of a topological space $X$ is a collection $\mathcal{U} = \{U_i\}_{i \in I}$ of open sets for some indexing set $I$ such that $\bigcup_{i \in I} U_i = X$. We assume a cover is finite and each $U_i$ is path-connected. Given a cover $\mathcal{U} = \{U_i\}_{i \in I}$ of $X$, let $\text{Nrv}(\mathcal{U})$ denote the nerve of the cover $\mathcal{U}$, defined as $\text{Nrv}(\mathcal{U}) = \{ \sigma \subseteq I \mid \bigcap_{i \in \sigma} U_i \neq \emptyset \}$.

In this paper, we are only concerned with the 1-nerve, that is, the 1-dimensional skeleton of the nerve, denoted as $\text{Nrv}^1(\mathcal{U})$. $\text{Nrv}^1(\mathcal{U})$ is a graph with nodes representing the elements of $\mathcal{U}$ and edges representing the pairs $(U_i, U_j)$ of $\mathcal{U}$ such that $U_i \cap U_j \neq \emptyset$.

Mapper. Given a continuous map $f : X \rightarrow \mathbb{R}$ where $f(X) \subseteq [a, b]$ is equipped with a cover $\mathcal{U} = \{U_i\}_{i \in I}$, we write $f(\mathcal{U})$ as the cover of $X$ obtained by considering the path-connected components of $f^{-1}(U_i)$ for each $i$.

Given such a function $f$, its mapper $M$ is defined to be the nerve of $f(\mathcal{U})$ [80], $M(f, \mathcal{U}) := \text{Nrv}(f(\mathcal{U})).$

In this paper, we start by considering a function $f : X \rightarrow \mathbb{R}$ and a cover $\mathcal{U}$ of $f(X) \subseteq [a, b]$ consisting of finitely many open intervals $\mathcal{U} = \{a_1, b_1\}, \ldots, \{a_n, b_n\}$. We then use the map $f$ to pull back the elements of $\mathcal{U}$ to obtain a cover of $X$ by consider path-connected components of $f^{-1}(a_i, b_i)$ for each $i$, denoted as $f^{-1}(\mathcal{U})$.

We are interested in the 1-nerve of such a cover on $X$, denoted as $M(f, \mathcal{U})$ for the remaining of the paper, when its dimension is implied.

There is a lot of flexibility in the construction of mapper. Fixing a space $X$, there are two primary parameters: a filter function $f$, and a cover $\mathcal{U}$ of its range space. Different filter functions can be used to capture different properties of the space $X$. That is why $f$ is thought of as a lens from which we view the space $X$.

On the other hand, fixing the map $f$ and choosing different covers for its range space can be used to obtain mappers at multiple resolutions.
We found 4 connected components in step (d) and so there are 4 nodes.

We now extend the general mapper construction by specifying the
weights. In the case when the graph $G$ has weights, the intrinsic
structure of data.

We start our pipeline with a graph $G = (V,E)$ that is part of the cover $\mathcal{U}$, we define a subgraph $G_i$ in $G$ as the one induced by the node set $f^{-1}(U_i)$. Specifically, let $V_i = \{v \in V | f(v) \in (a_i, b_i)\}$, and $G_i$ is a subgraph in $G$ induced by the
node set $V_i$. The connected components of $G_i$ form a cover $f^\circ(\mathcal{U})$ of $G$ and its 1-nerve is denoted as $M(G,f;\mathcal{U})$.

We explain this pipeline on Figure 2. We start with the input graph $G$ given in Figure 2 (a). We define a scalar function $f : V \rightarrow [a,b]$ on every node of $G$ as illustrated in Figure 2 (b). We then choose a cover $\mathcal{U}$ for $[a,b]$. In this case the choice of $\mathcal{U}$ consists of three intervals $U_A, U_B$ and $U_C$ as shown in Figure 2 (c). By a cover here we mean $[a,b] \subset U_A \cup U_B \cup U_C$. We then extract the connected subgraphs from $G$ that corresponds to each interval $U_A, U_B$ and $U_C$. Specifically, the connected subgraph that corresponds to the interval $U_A$ is the graph labeled $\alpha$ in Figure 2 (d). The interval $U_B$ corresponds to two connected subgraphs $\beta_1$ and $\beta_2$. Finally, the interval $U_C$ corresponds the connected subgraph $\gamma$. The final mapper graph $M(G,f;\mathcal{U})$ is shown in Figure 2 (e). Each connected component corresponds to a node in mapper graph.

We found 4 connected components in step (d) and so there are 4 nodes in the mapper graph. Edges in the final mapper graph are determined by the non-trivial intersection between the node sets of the connected components we found in step (d). For instance the subgraph $\gamma$ shares a node with the subgraph $\beta_1$ which corresponds to inserting an edge in the mapper graph between the nodes $\gamma$ and $\beta_1$. Note here that the overlap between the cover elements in necessary to have edges in the final mapper graph.

## 4 Parameter Exploration for Mapper Graphs

The mapper graph construction relies on the choice of two parameters: a filter function and a cover. We can also treat the exploration and manipulation of these parameters as a vehicle to study and summarize the intrinsic structure of data.

### 4.1 Filter Functions

An interesting open research problem is how to formulate filter functions beyond a best practice or a rule of thumb \[ \text{[7][8]} \]. In practice, height functions, distances from the barycentre of the space, surface curvature, integral geodesic distances and geodesic distances from a source point in the space have all been proposed as reasonable choices for filter functions \[ \text{[7]}. \]

In this section, we discuss three types of filter functions defined on the node set of a graph, as illustrated in Fig. 3 (a). Each function is chosen to reflect specific property of interest that is intrinsic to the structure of a graph. These choices are well-justified in a sense that they have been shown successful in a wide variety of applications beyond graphs.

### Average Geodesic Distance

Suppose a weighted graph $G = (V,E)$ is equipped with a geodesic distance metric $d$, that is, $d(u,v)$ measures the geodesic/graph distance between two nodes $u, v \in V$. $d$ can be computed by utilizing Dijkstra’s shortest path algorithm. The average geodesic distance, $\text{AGD} : V \rightarrow \mathbb{R}$, is given by

$$\text{AGD}(v) = \frac{1}{|V|} \sum_{u \in V} d(v,u).$$

This definition implies that the nodes near the center of the graph will likely have low function values, while points on the periphery will have high values. The $\text{AGD}$ function has been used extensively in shape analysis due to its desirable properties in detecting and reflecting symmetry \[ \text{[13]} \] based on how the function values are distributed. Therefore, the $\text{AGD}$ as a filter function is capturing the symmetrical properties of a graph, which are described by all or parts of the graph that are invariant to transformations such as reflection, rotation or scaling.

The mathematical notion of automorphism, in some sense, captures the symmetry of the space as it is a structural-preserving way of mapping a space to itself. More precisely, consider a graph $G$ as a metric space equipped with the geodesic distance, $(G,d)$. A bijection $T : V \rightarrow V$ is called an automorphism on $(G,d)$ if $d(u,v) = d(T(u), T(v))$ for every $u, v \in V$. Let $\text{Aut}(G)$ denote the group of automorphisms on $G$. A function $f : V \rightarrow \mathbb{R}$ is called an isometry invariant over $\text{Aut}(G)$ if for every $T \in \text{Aut}(G)$: $f \circ T = f$. The scalar $\text{AGD}$ is an isometry invariant scalar function. Indeed, let $T$ be on automorphism on $G$, then for every $v \in V$ we can verify:

![Fig. 2: The pipeline for a mapper construction on a graph. (1) A weighted graph $G(V,E)$. (2) A filter function $f : V \rightarrow \mathbb{R}$ with a range space $f(V) = [a,b]$. (3) A cover $\mathcal{U}$ of the range space is given by intervals $U_A$, $U_B$ and $U_C$. (4) The connected components of the subgraphs induced by the node sets $f^{-1}(U_A), f^{-1}(U_B)$ and $f^{-1}(U_C)$ form a cover of $G$, $f^\circ(\mathcal{U}) = \{\alpha, \beta_1, \beta_2, \gamma\}$. (5) The 1-nerve of $f^\circ(\mathcal{U})$ is the mapper graph, whose nodes represent the connected components and edges represent the non-empty intersections between the connected components.](image)

![Fig. 3: Examples of filter functions defined on the node set of a graph. (a) $\text{AGD}$ (orange). (b) Density estimation (green) with $\delta = 2$. (c) The Fiedler vector (purple) of the graph Laplacian. Darker colors represent lower function values.](image)
AGD(T(v)) = \frac{1}{|V|} \sum_{u \in V} d(T(u), T(v)) = \frac{1}{|V|} \sum_{u \in V} d(u, v) = AGD(v).

See Fig. 3(a) and Fig. 4(a) for examples of AGD on graphs.

Density Estimation. The density estimation function \([79]\) is given by

\[ D_\delta(v) = \sum_{u \in V} \exp\left(-\frac{d(u, v)^2}{\delta}\right), \]

where \(d(u, v)\) is the graph distance between two nodes in the graph and \(\delta > 0\). Since \(D_\delta\) is completely defined in terms of the distance \(d\), it is not hard to see that \(D_\delta\) is also isometry invariant.

\(D_\delta\) correlations negatively with AGD as it tends to take larger values on nodes which are close to the center, see Fig. 3(a-b) and Fig. 4(a-b) for examples.

Eigenfunctions of the graph Laplacian. Let \(G\) be an undirected, weighted graph with positive edge weights \(w: E \rightarrow \mathbb{R}\). Let \(C(G)\) be the vector space of all functions \(f: V \rightarrow \mathbb{R}\). The unnormalized Laplacian of the graph \(G\) is the linear operator \(L: C(G) \rightarrow C(G)\) defined by mapping \(f \in C(G)\) to \(L_f\), where

\[ (L_f)(v) = \sum_{u \in N(v)} w_{u,v} (f(v) - f(u)). \]

The eigenvectors of the Laplacian \(L\) form a rich family of scalar functions defined on \(G\) with many interesting geometric properties [57]. First, the gradient of the eigenfunctions of \(L\) tends to follow the overall shape of the data [57]; and these functions has been used in applications such as graph understand [78], segmentation [23], and spectral clustering [65]. Ordering the eigenvectors of \(L\) by the increasing value of their corresponding eigenvalues, we use eigenvectors of the second and third smallest eigenvalues of \(L\) as the filter functions, denoted as \(f_2\) and \(f_3\).

These vectors usually contain low frequency information about the graph, and they usually help retaining the shape of complex graphs. In particular, \(f_2\) is commonly referred to as the Fiedler vector [57] with desirable geometric properties [24]. For instance, the maximum and the minimum of the Fiedler vector tend to occur at points in the dataset with maximum geodesic distance [19], allowing it values to spread from one end of the graph following its "shape" to the other end. See Fig. 3(c) and Fig. 4(c) for examples.

4.1.1 Histogram of the Scalar Function
Understanding the distribution of the scalar values of \(f\) can be helpful in the mapper construction. Figure 5 shows an example of the histogram of the AGD on a graph. We will illustrate later how the visual information encoded in the histogram can be utilized to optimize the choice of the cover.

4.2 The Cover
Let \(f: V \rightarrow [a, b]\) be a scalar function defined on \(G\) and let cover \(\mathcal{U}\) be a cover of the interval \([a, b]\) consisting of the interval \(U_1 \cdots U_n\). We represent this cover visually by drawing rectangular boxes besides the histogram of the the scalar function as indicated in Figure 5.

As previously stated, mapper relies on a choice of cover for the interval \([a, b]\). The cover choice is rather flexible but also essential to achieve effective mapper visualization. To simplify the cover choice we normalize the scalar values of \(f\) from the interval \([a, b]\) to the unit interval \([0, 1]\) unless otherwise specified. This simplification does not yield any essential difference to the function \(f\). It merely simplifies our choices for the cover parameters that we will describe in this section.

In our pipeline the cover is constructed in two main steps: (1) the initial cover stage; and (2) interactive cover manipulation stage. We give a detailed description of these two stages below.

4.2.1 The Initial Cover Stage
The initial cover choice can be done by using the regular cover method. To obtain a regular cover of the interval \([a, b]\) we need two parameters: \(n \in \mathbb{Z}^+\) the cover resolution parameter, which is basically the number of cover elements, and \(\epsilon \in \mathbb{R}^+\) the overlap parameter, which indicates the amount of overlap between two consequent cover elements. To obtain this cover on the interval \([a, b]\), we start by splitting the interval into \(n\) subintervals \([c_1, c_2], [c_2, c_3], \ldots, [c_{n-1}, c_n]\) with equal length, such that \(c_1 = a\) and \(c_n = b\). The parameter \(\epsilon\) is then used to obtain the final cover \(\mathcal{U}(n, \epsilon) := (U_i := (c_i - \epsilon, c_{i+1} + \epsilon))_{i=1}^n\) for \([a, b]\). Choosing the parameters \(n\) and \(\epsilon\) will have a significant impact on the final mapper output. In Figure 6 we demonstrate examples of a graph \(G\) in Figure (a) with a given AGD filter defined on it. The values of the filter AGD were normalized to \([0, 1]\). Figures 6(b) to (f) show the result of the mapper graph of the graph \(G\) and the filter. In figures (b), (c) and (d) we fix the overlap parameter \(\epsilon\) and vary the cover resolution \(n\). On the other hand, if Figures (c), (e) and (f) we fix \(n\) and vary \(\epsilon\).

Choosing The Parameters of The Regular Cover. Our Method does
The initial cover stage is sufficient for most applications. However, in certain circumstances it is useful to manipulate the cover elements to aid the cover construction and achieve maximal control.

4.2.2 The Interactive Cover Manipulation Stage

The mapper graph is used to store the information one desires to extract from it. If the desired mapper graph $M$ summary is required to be small, then choosing the resolution $\epsilon$ tends to be within the range $[0.01, 0.3]$. Generally speaking, for large graphs and highly connected graphs we found that smaller values of $\epsilon$ give a more effective visualization. In this case, we found the range $[0.01, 0.3]$ to be sufficient for more purposes. On the other hand, for small graphs the value of $\epsilon$ tends to be within the range $[0.1, 0.3]$.

Using the Histogram to Help Choose the Regular Cover Parameters. The histogram of the scalar function can be used to decide the regular cover parameters as follows. Generally speaking, an evenly distributed histogram requires lower overlap $\epsilon$ parameters. When the histogram is not evenly distributed, as in the case of Figure 4, we usually choose higher overlap parameter $\epsilon$ to achieve the desired interconnection between the mapper nodes. This decision is also made with consideration of the size of the original graph.

4.2.2 The Interactive Cover Manipulation Stage

The initial cover stage is sufficient for most applications. However, in certain circumstances it is useful to manipulate the cover elements individually to achieve a different and more desirable mapper graph output. For this purpose we developed an interactive cover manipulation interface to aid the cover construction and achieve maximal control over the shape of the output. The user can select one interval in the cover and manipulate its ends dynamically in the interface.

Given an interval $U_i = [a_i, b_i]$ in the cover $\mathcal{U}$, our interactive cover interface allows the following functionalities on $U_i$:

- Shrinking the interval $U_i$ to obtain a new interval $U_i'$.\[ \text{Shrinking} \]
- Expanding interval $U_i$ to obtain a new interval $U_i'$.\[ \text{Expanding} \]
- Shifting interval $U_i$ to obtain a new interval $U_i'$.\[ \text{Shifting} \]

These three cases are illustrated in Figures 7. We want analyze the change that occur on a mapper graph $M(G_i, \mathcal{U})$ when a certain interval $U_i = [a_i, b_i] \in \mathcal{U}$ changes to a new interval $U_i'$ from the above three cases. In other words, given a cover element $U_i$ in $\mathcal{U}$ we want to be able to compute the mapper graph when $U_i$ changes to another cover element $U_i'$ efficiently assuming that all other mapper parameters are fixed.

Recall from Section 3 that for a given interval $[a_i, b_i]$, the graph $G_i$ is the subgraph of $G$ that is mapped to $[a_i, b_i]$ via $f$. As we perform one of the operations, expansion, shrinking or shifting, on the interval $[a_i, b_i]$ the graph $G_i$ changes to a new graph $G_i'$. We illustrate the possible changes that occur on the mapper graph by demonstrating the changes that occur on the graph $G_i$. As we change the interval $U_i$, the connected components of $G_i$ change by merging with other components, splitting to new components, new connected components might appear and existing ones might disappearing.

Below we give a detailed description of how the above three interval operations affect the mapper graph.

The Interval $U_i$ Expands. This occur when the interval $U_i = [a_i, b_i]$ changes to an interval $U_i' = [a_i - \epsilon, b_i]$ or $U_i' = [a_i, b_i + \epsilon]$ for some $\epsilon > 0$. See Figure 7(a) for an illustrative example.

Since $U_i \subset U_i'$ then $f^{-1}(U_i) \subset f^{-1}(U_i')$ and hence the graph $G_i$ is a subgraph of $G_i'$. In other words, due to the expansion of the interval $U_i$ more nodes and edges from the original graph $G$ get added to form the graph $G_i'$. This means that each existing connected component of $G_i$ will remain connected in $G_i'$ potentially with more nodes and edges added to it. However, as a result of adding more nodes and edges, the following topological changes may occur to the connected components of $G_i$:

- Multiple existing disjoint connected components of $G_i$ might merge in $G_i'$ to produce a single connected component. This means that the corresponding nodes in the mapper graph will also get merged. See Figure 7(a) and (b).
- As a result of the expansion new connected component might also appear. This will result in the creation of a new node in the mapper graph. See Figure 7(a) and (b).

The Interval $U_i$ Shrinks. This occurs when the interval $U_i = [a_i, b_i]$ changes to an interval $U_i' = [a_i + \epsilon, b_i]$ or $U_i' = [a_i, b_i - \epsilon]$. This is merely the dual case of the previous case. Namely, as we shrink the interval $U_i$ into the interval $U_i'$ the graph $G_i'$ will be a subgraph of
the graph $G_i$. As a consequence an existing connected component in $G_i$ might split into multiple connected components in $G'_i$. On the level of the mapper graph, the corresponding node splits into multiple nodes. Moreover, certain connected components in $G_i$ might disappear in $G'_i$ which translates into deleting the corresponding node in the mapper graph.

The Interval $U_i$ Shifts. This occurs when the interval $U_i = [a_i, b_i]$ changes to an interval $U'_i = [a_i + \epsilon, b_i + \epsilon]$. This case can be considered as a combination of the sequence of expansion and shrinking of intervals as follows. If the shift is positive (i.e., upwards), the first step we change the interval $[a_i, b_i]$ to the interval $[a_i, b_i + \epsilon]$. In the second step we change interval $[a_i, b_i + \epsilon]$ to the interval $[a_i + \epsilon, b_i + \epsilon]$. For a negative (i.e., downward) shift, we expand $[a_i - \epsilon, b_i]$ and then shrink $[a_i - \epsilon, b_i - \epsilon]$. In both steps we are back to cases of expansion and shrinking discussed earlier. See Figure 2(c) and (d) for an illustrative example.

Using this capability, a given initial regular cover can be adjusted to realize a desirable "feature" in the data. For instance, Figure 24(c) shows the circular feature in the mapper graph, capturing the corresponding "circular feature" in the original graph. The cover was interactively manipulated, starting from the cover shown in Figure 24(b). It is also useful to mention that the histogram of filter is useful in deciding the location of each individual interval.

5 Visual Design and Interaction

The goal of our design is to enable exploration of the structure of the graph. This is done by providing a linked-view interface between the original graph, a graph summary in the form of the mapper graph, and an interactive cover designer component that allows for customization of the mapper graph.

5.1 Graph Drawing

Graph Layout Method. For both the original graph and the mapper graph, we utilized a force-directed layout [44] with the Barnes-Hut optimization for repulsive force [4]. This approach was chosen for its interactive nature. However, our approach is ultimately agnostic of the underlying graph layout algorithm, and different layouts may improve the presentation of certain graphs.

Node Coloring. As a part of the input for mapper we are given the filter function $f : V \rightarrow [a, b]$. The filter function choice provides the hue: red for AGD, green for density, and purple for eigenfunctions. The scalar value is mapped to the saturation value of the color. This color scheme is used directly for the nodes of the original graph, as illustrated in Figure 5. Moreover, this map is also used to color the nodes of the mapper graph by taking the average of the filter function values in the connected component. For a given connected component, $C_v$, the average is $\frac{1}{|C_v|}\sum_{u \in C_v} f(u)$.

Node Size. For the mapper graph, the size of the node is chosen to be proportional to the cardinality of the associated connected component, in other words |$C_u$|.

Edge Thickness. For both graphs, edge thickness is drawn proportional to edge weight. For the mapper graph, recall that an edge $[u, v]$ is determined by checking the intersection between $C_u$ and $C_v$. For this reason, the edge weight, and thus thickness, is drawn proportional to the set intersection size, $C_u \cap C_v$.

5.2 Interactive Cover Designer

The interactive cover designer consists of 2 main elements. First on the left is the histogram of filter function $f : V \rightarrow [a, b]$. This provides intuition as to how function values occur frequently and thus may be interesting to investigate further. The second element is the cover to the right. Each box represents a single cover element. The vertical position is its function value range, and the horizontal position is selected using the first-fit bin packing algorithm to minimize space consumption. As the cover elements are moved, expanded, or shrunk, the mapper graph visualization dynamically adds and removes nodes and edges based upon the approach outlined in Section 4.2.2.

5.3 Correlating the Substructures to the Original Graph

Having the mapper summary it is important to be able to extract the correspondence between the original data and the summary represented by mapper. All mapper components represented in our interface and can be selected dynamically to understand the structure encoded in the mapper graph. As components are selected, highlighting establishes the correspondence between the cover, the mapper graph elements, and the original graph. We provide the user with three mechanisms for exploration: node selection, edge selection, and cover element selection.

Node Selection. Each node in the mapper graph corresponds to a connected component from the original graph. Selecting such a node, our interface recovers and highlights the original connected component from the graph. Furthermore, the cover element which generated the mapper graph node is additionally highlighted. For all selection examples, the color hues are chosen to be complimentary to the hue used for the filter function to maximize visual difference. Figure 8 illustrates an example of node selection in a given mapper graph.
Fig. 9: (a-b) Selecting two different nodes in the mapper graph. Selecting a node (blue) in the mapper graph highlights the corresponding nodes in the original graph (also in blue). (c) Selecting an edge in the mapper graph (blue) selects the two nodes in the mapper graph that are attached to this node as well as the nodes in the original graph that are attached to the edge.

Fig. 10: Selecting an interval from the cover corresponds to selecting the nodes in the mapper graph that corresponds to that interval, as well as the nodes in the original graph. Figures (a-c) show the effect of selecting intervals and how it triggers the selection of the mapper graph and the original graph nodes.

6 Spectral Clustering and Mapper

In classical spectral clustering the eigenfunctions of the graph Laplacian can be used to obtain segmentation algorithms of the underlying graph. For instance the Fiedler’s vector $f_2$ can be used to bi-partition the graph $G$ into two parts $C_1 = \{v \in V | f_2(v) > 0\}$ and $C_2 = \{v \in V | f_2(v) \leq 0\}$. Spectral partition is well-studied, and its justification can be found in many places in the literature [67].

The mapper construction that we presented here can be considered, in the trivial case, as a generalization for spectral clustering. For instance, the segmentation induced by the Fiedler’s vector $f_2$ is nothing more than the mapper graph $M(f_2, G, \mathcal{U}')$ where $\mathcal{U}'$ is the regular cover $U(2,0)$. Higher order eigenfunctions of the Laplacian have also been used for segmentation purposes in an analogous fashion [77].

The eigenfunctions of the graph Laplacian can be used to obtain an approximation for the graph mincut problem [86]. From this, edges in the mapper graph using the eigenfunction of the unnormalized Laplacian can be interpreted as an approximation of the relations between the graph segments obtained from the mincut problem. From this perspective mapper does not only provide a generalization of spectral clustering but also provide the connection between the clusters induced using these spectral techniques.

Figure 11 illustrates the mapper graph of the USAIR 97 graph with the filter function $f_3$ and the regular cover $U(5,0)$. Of course the mapper graph here is trivial and has merely 5 nodes representing the 5 clusters captured by $f_3$. Later in the next section, we return to this example and demonstrate how mapper can be used to give insights about the relation among these clusters, something classical spectral clustering does not provide directly.

Fig. 11: The clustering induced by the eigenfunction $f_3$ of the Laplacian on a graph can be seen as a special case of the mapper graph construction. Here the chosen cover is $U(5,0)$ gives 5 clusters of the graphs, two of them are highlighted in blue. These are exactly the same clusters obtained by using the spectral clustering on $G$ using $f_3$.

7 Results

In this section we evaluate our approach by examining mapper graph on synthetic and real datasets.

7.1 Mapper on Synthetic Data

We validate our method by testing the output of the mapper graph on 20 synthetic datasets. All our graphs are generated using NetworkX [40]. Table 1 shows synthetic graphs and their corresponding mapper examples results. In each example a certain structure is emphasized via our choice of the scalar function and the cover. This structure could be symmetry, as in Figures 13, 18 and 22 or the overall shape of the graph, as in Figures 14, 17 and 23. One can observe that original graphs in Figures 12 and 17 have circular shape so here we choose the Fiedler’s vector as our choice of filter because we wanted a scalar function that can vary from one end of the graph to the other end. The mapper graph in Figure 17 looks interesting from our perspective because the original graph appears as a torus mesh and the mapper graph looks like the Reeb graph one usually have when computing Reeb graph on a torus mesh. Mapper also seems to capture the dual structure of some of graph examples such as the cases in [22] and [73].

It is worth mentioning that while Table 1 illustrates a single choice of a specific scalar function and a cover. Other choices could also be valid depending on the data and the context of the summary one wants to obtain from the graph.

7.2 Map of Science

The map of science graph [9]. Figure 24 (a), consists of 554 nodes and 2276 edges. Nodes in this graph represent specialties with major scientific disciplines and edges represent co-authorship of publications between those specialties. Figure 24 shows the graph with its nodes colored according to the major scientific disciplines of the node.

Since this graph does not seem to exhibit obvious symmetry, our choice for the mapper graph scalar function was the eigenfunctions of the Laplacian. As mentioned earlier, the smallest eigenfunctions of the graph Laplacian can help retaining the shape of the graph. The third smallest eigenfunction of the graph Laplacian $f_3$ was selected as it gave it retained the shape of the graph better than the Fiedler vector $f_2$.

Figure 24 (b) shows the mapper graph calculated on the map of science graph using $f_3$ with the regular cover parameters $n = 10$ and $\varepsilon = 0.1$. One can see from the figure that the mapper graph clearly preserves the overall structure and shape of the original graph. Moreover, the indicated mapper graph nodes capture certain super clusters in the original graph.

Recall that the eigenfunctions of the unnormalized Laplacian can be used to approximate the graph mincut problem. Moreover the mincut output usually prefers isolated clusters in the graph [86]. Given the interpretation for the mapper graph using the eigenfunction of the Laplacian we provide in Section 6 the clusters shown in Figure 24 (b)
are justified. However, it maybe desirable to obtain a mapper graph summary that captures more closely the shape and categories of the original graph.

To illustrate that the mapper graph can be used to obtain a better representation of the original graph, we utilize the interactive cover capabilities presented in Section 5.2. The graph is shown in Figure 24 (c). In the mapper graph in Figure 24 (c) the nodes are circled to highlight the majority scientific discipline from the underlying cluster. For instance the nodes that are labeled 1 in the mapper graph represent the Humanities nodes in the original graph.

Figure 24 (c) indicates how the mapper graph in this instance gives a small introductory summary of the original graph. This summary is indicated in both the clusters and the relationship among those clusters. For instance node 9 and node 5 are branching nodes, namely these
are nodes where the mapper graph changes its topology. Inspecting communities 8 and 6 we can see that these communities represent Chemistry and Biology. These two field merge at the node 5, which represents medical science and infectious diseases.

7.3 USAIR 97

The previous example illustrates the natural interpretation of the mapper graph nodes as clusters of the original graph. In Figures 25 we illustrate how the nodes and edges of the mapper graph show natural connection between clusters. The USAIR 97 graph consists of 332 nodes and 2126 edges [6]. The nodes represent airport and the edges represent the connection between the airports. For mapper graph setting we use the eigenfunction of the Laplacian $f_3$ with interactive cover setting as shown in Figure 11. Recall that we illustrated in Figure 1 on the same graph how the eigenfunction we obtained the edge $[2,3]$. In Figure 25 (a), the cluster 1 in cluster 2, which contains Anchorage international. (b) The edge $[2,3]$ than $[1,2]$. In Figure 25 (a), the cluster 1, which contains Anchorage international, which is contained in the original graph nodes that represent the mapper graph edge $[5,6]$.

Finally, the mapper graph node 6 corresponds to the cluster $C_6$, shown in Figure 1 represents a peripheral cluster that is far away from the main bulk of the graph. The mapper graph node 6 is represented mainly by the Guam international airport. In order to pass from any airport in cluster $C_6$ to the any airport in the main central bulk representing by mapper graph node 5 one must pass from the Honolulu International, which is contained in the original graph nodes that represent the mapper graph edge $[5,6]$.

Fig. 24: (a) Circle of Science categories. (b) Circle of Science mapper graph using the regular cover $U(10, 0.1)$ and the third smallest eigenfunction we obtained the mapper graph shown on the top. The shape preservation for the mapper graph of the original graph is indicated in the layout of the mapper graph and the original one. Some mapper graph nodes naturally capture clusters in the graph. (c) mapper graph using the interactive cover achieves better clustering quality and shape summary.

Fig. 25: Selection edge in the mapper graph and the corresponding clusters in the original USAIR 97 graph. (a) The edge $[1,2]$ is selected indicating the connection between the cluster 1 that contains Bethel and the cluster 2, which contains Anchorage international. (b) The edge mapper graph $[2,3]$ is selected and highlights in blue in the nodes in the original graph. In order to go from the cluster that contains Juneau International to the any of the airports in light blue one must pass through Anchorage international, which represents the main airport in the mapper graph edge $[2,3]$.
8 Conclusion

We have presented a topological data analysis approach to generate a summary for a graph using the mapper construction. Our method is effective at finding clusters or communities in a graph and the relations among these clusters. The strength of our construction lies by its flexibility in the being able to capture the structure of the underlying graph on multiple scales and using different topological and symmetrical properties. Finally, in the future, we would like to explore other potential applications for mapper graph. Other than graph exploration, mapper graph can be consider as a skeleton for the underlying graph, which could be used for many other tasks, such as graph layout.

References


